```
/* pre-processing steps for FPP
*/

options notes mprint validvarname=upcase noovp;

%let FPPCODE = %sysget(FPPCODE);
%let DSBASE=%sysget(DSBASE);
%include "&FPPCODE/utilities/macros.macro";

%include "&FPPCODE/pop/optimizer/getenv.macro";

%put getting environment;

%getenv(makenew=1, inpf=0);

%put includes;
%include "&FPPCODE/utilities/import_csv.macro";
%include "&FPPCODE/utilities/dbutil.macro";

%include "&FPPCODE/pop/optimizer/target_adjuster.macro";
%include "&FPPCODE/pop/optimizer/discopto.macro";
%include "&FPPCODE/pop/optimizer/mk_pfs.macro";
%include "&FPPCODE/pop/optimizer/pop2_opto.macro";
%include "&FPPCODE/pop/optimizer/fppalpha.macro";
%include "&FPPCODE/pop/optimizer/product.macro";

%include "&FPPCODE/fpp/ld_fprs.macro";
%include "&FPPCODE/fpp/pre_optimize.macro";

%put loading FPRS data;

%ld_fprs;

%put get subsets of reference data and set up to run;

%pre_cons;

%put set up portfolios;

%mk_pfs(fixedassets=1);

%put compute alpha for funds;

%fppalpha(in_tv=product.asset_tv, ast_dtr=product.dtr, idx_dtt=product.idx_dtt, mdl_ou
t=product.alpha);

%put set up asset_tv and opop;

data product.asset_tv;
   merge product.asset_tv product.alpha(keep=ticker alpha pvaluem);
   by ticker;
   run;

data product.opop;
   lambda=5000;
   kappa=.6;
   linear='PVALUEM';
   minval=.05;
   whole = 1;
   run;
```

Exhibit A

BEST AVAILABLE COPY

pfopt.sas


```
options nonotes nomprint validvarname=upcase noovp;

%let FPPCODE = %sysget(FPPCODE);
%let DSBASE=%sysget(DSBASE);
%include "&FPPCODE/utilities/macros.macro";
%include "&FPPCODE/utilities/dbutil.macro";

%include "&FPPCODE/pop/optimizer/getenv.macro";
%include "&FPPCODE/pop/optimizer/target_adjuster.macro";
%include "&FPPCODE/pop/optimizer/discopto.macro";
%include "&FPPCODE/pop/optimizer/pop2_opto.macro";
%include "&FPPCODE/pop/optimizer/mk_pfs.macro";
%include "&FPPCODE/pop/optimizer/ctw2whole_pct.macro";

%include "&FPPCODE/fpp/special_cases.macro";

%getenv(makenew=1, inpf=1);

%opto_pf;
```

```
   options nonotes nomprint validvarname=upcase noovp;

%let FPPCODE = %sysget(FPPCODE);
%let DSBASE=%sysget(DSBASE);
%include "&FPPCODE/utilities/macros.macro";
%include "&FPPCODE/utilities/dbutil.macro";


%include "&FPPCODE/pop/optimizer/getenv.macro";
%include "&FPPCODE/pop/optimizer/target_adjuster.macro";
%include "&FPPCODE/pop/optimizer/mk_pfs.macro";
%include "&FPPCODE/pop/optimizer/pop2_post.macro";


%getenv;

%pop2_postprocess(prune=1);



   proc sql;
   create table product.all_mdls as
      select c.pfname, a.ticker,d.fprs_id,round(a.wgt,.000000001) as wgt
      from product.mdl_ctw a, product.pfmaster b, product.pfmaster c, product.plast d
      where a.cticker=b.pfid
      and b.cticker=c.proxyplan
      and c.risklev=b.risklev
      and c.w1=b.w1
      and d.cticker=c.cticker
      and d.ticker=a.ticker
      and round(a.wgt,.000000001) ne 0
      order by c.pfname;

   proc sql;
   create table product.all_terr as
      select c.pfid as ticker, a.trackerr*100 as trackerr
      from product.mdl_terr a, product.pfmaster b, product.pfmaster c
      where a.ticker=b.pfid
      and b.cticker=c.proxyplan
      and c.risklev=b.risklev
      and c.w1=b.w1
      order by ticker;

   proc sql;
   create table product.all_target as
      select c.pfname, a.ticker,round(a.wgt,.000000001) as wgt
      from product.mdl_target a, product.pfmaster b, product.pfmaster c
      where a.cticker=b.pfid
      and b.cticker=c.proxyplan
      and c.risklev=b.risklev
      and c.w1=b.w1
      order by c.pfname;
```

```
   options nonotes nomprint validvarname=upcase noovp;

%let FPPCODE = %sysget(FPPCODE);
%let DSBASE=%sysget(DSBASE);
%include "&FPPCODE/utilities/macros.macro";


%include "&FPPCODE/fpp/getenv.macro";

%include "&FPPCODE/pop/optimizer/target_adjuster.macro";
%include "&FPPCODE/pop/optimizer/discopto.macro";

%include "&FPPCODE/fpp/ld_fprs.macro";
%include "&FPPCODE/utilities/import_csv.macro";
%include "&FPPCODE/fpp/pre_optimize.macro";
%include "&FPPCODE/fpp/mk_pfs.macro";
%include "&FPPCODE/fpp/hc_atype.sas";
%include "&FPPCODE/fpp/fppalpha.macro";
%include "&FPPCODE/fpp/optimize_pf.macro";
%include "&FPPCODE/pop/optimizer/ctw2whole_pct.macro";
%include "&FPPCODE/utilities/dbutil.macro";

%macro assemble_pf;
   /* assemble all model portfolios into a single ctw */

   %global pfid;

   %getenv;

   %pfma(pma=pf,nma=pn);

   %local i;
   %do i = 1 %to &pf0;
      %let pfid = &&pf&i;
      %let pfname = &&pn&i;
      %if %sysfunc(LIBNAME(pf,&fppmast/pf/&pfname)) eq 0 %then %do;
         %if %sysfunc(EXIST(pf.mdl_ctw)) %then %do;
            %put &pfid &pfname OK;
            %dsappend(i=&i,in_ds=pf.mdl_ctw,out_ds=fppmast.mdl_ctw,same=1);
         %end;
         %else %put &pfid &pfname NO mdl_ctw;
      %end;
      %else %put &pfid &pfname NO directory;
   %end;

%mend;

%macro summ_pf(in_ctw=,out_dtr=,out_terr=);

   /* get benchmarks for risk levels */

   %ctwret(in_ctw=fppmast.benches,in_dtt=fppmast.fpp_dtt,out_dtt=bench_dtt);

   data fppmast.fpp_dtt;
   merge fppmast.fpp_dtt bench_dtt;
   by date;
   run;

   /* compute tracking error for each risk level separately */

   %local i;
   %do i = 1 %to 6;

      data _levctw;
      set &in_ctw(where=(risklev=&i));
      run;
      %terr_ctw(in_ctw=_levctw,bench=_BENCH&i,in_dtt=fppmast.fpp_dtt,out_ds=terr,perye
ar=12,out_dtt=ctw_dtt,rebal=1);
      %dtt2dtr(in_dtt=ctw_dtt,out_dtr=ctw_dtr);
      %dsappend(i=&i,in_ds=terr,out_ds=&out_terr,same=1);
      %dsappend(i=&i,in_ds=ctw_dtr,out_ds=&out_dtr,same=1);
   %end;
   proc sort data=&out_dtr;
   by ticker;
   run;
   proc sort data=&out_terr;
   by ticker;
   run;


%mend;

data fppmast.benches;
   length cticker $8 ticker $8;
   input cticker ticker $ wgt;
cards;
_BENCH1 _W5000 .85
_BENCH1 _BOND 0
_BENCH1 _CASH 0
_BENCH1 _EAFE .15
_BENCH2 _W5000  .70
_BENCH2 _BOND   .15
_BENCH2 _CASH    0
_BENCH2 _EAFE .15
_BENCH3 _W5000 .60
_BENCH3 _BOND .25
_BENCH3 _CASH .05
_BENCH3 _EAFE .10
_BENCH4 _W5000 .45
_BENCH4 _BOND .40
_BENCH4 _CASH .10
_BENCH4 _EAFE .05
_BENCH5 _W5000 .20
_BENCH5 _BOND .30
_BENCH5 _CASH .50
_BENCH5 _EAFE 0
_BENCH6 _W5000 0
_BENCH6 _BOND 0
_BENCH6 _CASH 1
_BENCH6 _EAFE 0
;
run;

%assemble_pf;
```

```
%macro alphat(in_tv=, atype=, idx_dtt=, ast_dtr=, factors=, out_mdl=);
/* compute alphas for a given asset type and factors
in_tv(ticker,atype) reference data
atype - asset type to filter &in_tv on
idx_dtt(date,&factors) - factor returns
ast_dtr(date,ticker,return) asset returns
factors - list of factors in regression
out_mdl(ticker,rmse,alpha,_rsq_, _adjrsq_, pvalue, &factors) output model
*/
   %let factcl = %l2cl(list=&factors);
   proc sql noprint;
   select count(*)
      into :_periods
      from &idx_dtt;

   proc sql;
   create table _lretx as
      select a.ticker, a.return-c._cash_ as return, &factcl
      from &ast_dtr a, &in_tv b, &idx_dtt c
      where a.ticker=b.ticker
      and a.date = c.date
      and b.atype = "&atype"
      order by a.ticker;

   proc reg data=_lretx outest=&out_mdl noprint;
        model return = &factors / ADJRSQ MSE ;
        by ticker;
   run;
   data &out_mdl(keep=ticker rmse alpha &factors _RSQ_ _ADJRSQ_ pvaluem);
   set &out_mdl(rename=(_rmse_=RMSE intercept=ALPHA));
   if rmse eq 0 then alpha = 0;
   if rmse eq 0 then aph=0;
   else aph=(alpha/rmse)*sqrt(&_periods);
   pvaluem = -probt(aph, &_periods-1);
   run;

%mend;

%macro fppalpha(in_tv=, ast_dtr=, idx_dtt=, mdl_out=);
 /* calculates alphas for all assets
in_tv(ticker,atype,...) assets to compute alphas for. atype indicates which alpha mode
l to use
idx_dtt(date,ticker1,...) return series of indexes
ast_dtr(date,ticker,return) returns series of assets
mdl_out(ticker,alpha,pvalue,...) alpha and pvalue of assets
*/

   data alphadtt;
   set &idx_dtt;
   _LRGSML = _R1000 - _R2000;
   _GROVAL = _R3000G - _R3000V;
   _EXCSTK = _R3000 - _CASH_;
   _EXCBND = _BOND - _CASH_;
        _CREDIT = _ML_HYM - _LB_TR;
        _DURL = _SB_10TR - _SB_TR;
        _DURS = _SB_1YTR - _SB_TR;
        _XMBS = _LB_MBS - _BOND;
        _XFARE = _MS_FARE - _CASH_  ;
        _XEURO = _MS_EURO - _CASH_  ;
        _XAUST = _MS_AUST - _CASH_  ;
        _XNA = _MS_NA - _CASH_  ;
        _XLAM = _IF_LAM - _CASH_  ;
        _XUSD = _USDTW - _CASH_  ;
        _XMUNI = _LB_MUNI - _CASH_  ;
        _XDURMU = _LB_10MU - _LB_14MU;
   run;

   %alphat(in_tv=&in_tv,atype=EQUITY ,idx_dtt=alphadtt, ast_dtr=&ast_dtr, factors=_LRG
SML _GROVAL _EXCSTK _EXCBND,out_mdl=eqalph) ;
   %alphat(in_tv=&in_tv,atype=BOND   ,idx_dtt=alphadtt, ast_dtr=&ast_dtr, factors=_EXC
BND _CREDIT _DURL _DURS _XMBS,out_mdl=bndalph) ;
/*    %alphat(in_tv=&in_tv,atype=MUNI   ,idx_dtt=alphadtt, ast_dtr=&ast_dtr, factors=_
XMUNI _XDURMU,out_mdl=munialph) ;*/
   %alphat(in_tv=&in_tv,atype=FOREIGN,idx_dtt=alphadtt, ast_dtr=&ast_dtr, factors=_XFA
RE _XEURO _XAUST _XNA _XLAM _XUSD,out_mdl=frnalph) ;

   data &mdl_out;
   set eqalph bndalph /* munialph */ frnalph end=eof;
   output;
   if eof then do;
      ticker='_CASH_';
      pvaluem=.5;
      output;
      ticker='_GIC_';
      pvaluem=.5;
      output;
   end;
   run;

   proc sort data=&mdl_out;
   by ticker;
   run;


%mend;
```

ld_fprs.macro


```
%macro ld_fprs;
/* load fprs data:
input:
&channel/Plan_Options_Monthly_Return.csv
&channel/Plan_Options_Supplement.csv
&channel/Plan_Options_Master.csv
&channel/cstk401k.csv
&channel/hardCode.csv
output:
product.fprs_dtt
product.fprs_tv
product.rawplast
product.fixedast
product.rawproxy
*/

   /* input returns */
   %imp_csv(&channel/Plan_Options_Monthly_Return.csv, _sret, 1, %str(,) );

   data _sret;
        length ticker $8;
        set _sret;
        ticker = "_"||FPRS_ID;
   run;

   proc transpose data=_sret(drop=FPRS_ID f_name ) out=_stret(drop=_label_) name=DATES
;
        id ticker;
   run;

   data product.fprs_dtt(drop=dates);
   format date 8.;
   set _stret;
   array _x _numeric_;
   do over _x;
      _x = _x / 100;
   end;
   date = input(substr(dates,2),10.);
   run;

   /* input characteristics */

   %imp_csv( &channel/Plan_Options_Supplement.csv, _sup, 1 , %str(,) );

   data product.fprs_tv;
   format ticker $8.;
   set _sup;
   array _x stocks bonds cash other foreign;
   ticker = compress("_"||fprs_id);
   do over _x;
      if _x eq . then _x = 0;
   end;
   run;

   /* input plans/assets */

   %imp_csv( &channel/Plan_Options_Master.csv, product.rawplast, 1, %str(,) );

   /* input fixed assets (company stock data) */

   %imp_csv( &channel/cstk401k.csv, product.fixedast, 0, %str(,) );
   data product.fixedast;
   format cticker ticker $8.;
   set product.fixedast;
   cticker=plan;
   wgt = weight/100;
   run;

   /* input proxy asset mapping rules */

   %imp_csv(&channel/hardCode.csv , product.rawproxy , 0, %str(,) );

%mend;
```

```
%macro setpf(pfn);
   x "setenv PFNAME &PFN";
   %getenv(inpf=1);
%mend;


%macro mk_pfs(fa_ds=, pf_ds=, in_portf=, risklevs=6);
/* make portfolios
input:
fa_ds(cticker,ticker,wgt) optional fixed assets: a portfolio for each cticker X ticker
-wgt (i.e. company stock levels)
in_portf(cticker,proxyplan) portfolios to optimize
also one for each cticker not in &fa_ds
output:
pf_ds(pfid, pfname, pflname,cticker,risklev,proxyplan,t1..tn,w1..wn) the pfmaster
*/

   %global maxfa;
   %local i;

   %let maxfa = 0;

  proc sql;
   create table &pf_ds as
      select distinct cticker,proxyplan
      from &in_portf
      order by cticker;

   %if &fa_ds ne %then %do;
      /* this is map from FA1,... to tickers for each plan(cticker) */

      proc sql;
      create table fa_map as
         select cticker,ticker as fatick,count(*) as lvl
         from &fa_ds a, product.plans  b
         where a.cticker=b.plan
         group by cticker, ticker
         order by cticker, ticker;

      data fa_map(drop=tickno maxfa);
      set fa_map end=eof;
      by cticker;
      retain tickno 0 maxfa;
      format faid $8.;
      if first.cticker then tickno=0;
      tickno = tickno + 1;
      if maxfa < tickno then maxfa=tickno;
      faid = 'FA'||put(tickno,1.);
      if eof then call symput("MAXFA",compress(put(maxfa,3.)));
      run;

      proc sql noprint;
      create table fat as
      select a.cticker, b.faid, a.ticker, a.wgt
      from &fa_ds a, fa_map b
      where a.cticker=b.cticker
      and a.ticker = b.fatick
      order by a.cticker,b.faid;


      data _pf_ds(drop=faid rename=(ticker=t1 wgt=w1));
      set fat(where=(faid='FA1'));
      run;

      %do i = 2 %to &maxfa;
         data fat2;
         set fat(where=(faid="FA&i"));
         run;

         proc sql noprint;
         create table _pf_ds as
            select a.*,b.ticker as t&i, b.wgt as w&i
            from _pf_ds a left join fat2 b on  a.cticker=b.cticker
            order by a.cticker;
      %end;


      data &pf_ds;
      merge _pf_ds &pf_ds;
      by cticker;
      run;

   %end;

   data &pf_ds;
   set &pf_ds;
   %do i = 1 %to &risklevs;
      risklev = &i;
      output;
   %end;
   run;

   proc sort data=&pf_ds;
   by cticker risklev  %do i = 1 %to &maxfa; w&i %end;;
   run;


   data &pf_ds;
   format pfid $8. pfname $20. pflname $40.;
   set &pf_ds;
   pfid = "PF"||put(_n_,Z6.);
   pfname = trim(cticker)||'/'||put(risklev,1.);
   pflname = trim(pfname);
   %do i = 1 %to &maxfa;
      if w&i = . then w&i = 0;
      pfname = trim(pfname)||'/'||put(w&i*100,z2.);
      pflname = trim(pflname)||'/'||trim(t&i)||'/'||put(w&i*100,z2.);
   %end;
   run;

   proc sql noprint;
   create index pfname on &pf_ds(pfname);

%mend;


%macro pfid2mac(pfid=);
/* pfid to macros cticker risklev stktck1-stktckn stkwgt1-stkwgtn pfname pflname
*/

   /* get maximum number of fixed assets in a plan */
   %global maxfa cticker risklev pfname pflname;
   %local i;
   proc contents data=product.pfmaster(drop=pfid cticker risklev)
   out=_m1 short noprint;
   run;
   proc sql noprint;
   select count(*)/2 into :maxfa
      from _m1;
```

```
   %do i = 0 %to &maxfa;
      %global stkwgt&i stktck&i;
   %end;

   %let stkwgt1 = 00;
   %let stktck1 = ;

   data _null_;
   set product.pfmaster(where=(pfid="&pfid")) end=eof;
   call symput("CTICKER", cticker);
   call symput("RISKLEV", compress(put(risklev,2.)));
   stklev0=0;
   %do i = 1 %to &maxfa;
      if t&i ne '' then do;
         stklev0 = stklev0 + 1;
         call symput("STKWGT&i",compress(put(w&i*100,z2.)));
         call symput("STKTCK&i",compress(t&i));
      end;
   %end;
   if eof then call symput("STKWGT0",compress(put(stklev0,2.)));
   run;
   %let stktck0 = &stkwgt0;
   %let pfname = &cticker/&risklev;
   %let pflname=&pfname;

   %do i = 1 %to &maxfa;
      %let pfname = &pfname/&&stkwgt&i;
      %let pflname = &pflname/&&stktck&i/&&stkwgt&i;
   %end;

%mend;

%macro pfma(pma=,nma=);

   %global &pma.0 &nma.0;

   data _master;
   set product.pfmaster(where=(proxyplan=cticker));
   run;


   proc contents data=_master(drop=pfid pfname pflname cticker risklev proxyplan )
   out=_m1 short noprint;
   run;
   proc sql noprint;
   select count(*)/2 into :maxfa
      from _m1;

   %ds2ma(in_ds=_master, var=pfname, ma=&nma, uniq=0);
   %ds2ma(in_ds=_master, var=pfid, ma=&pma, uniq=0);

%mend;


%macro pfnam2mac(pfname=);
/* pfname to macros cticker risklev stktck1-stktckn stkwgt1-stkwgtn pfid pflname
*/

   %global pfid pflname cticker risklev maxfa sumstklev ;
   %local i j p im2;

   data _null_;
   pfname = translate("&pfname",' ','/');
   call symput("P", pfname);
   run;

   %put pfname=&pfname, p=&p;

   %l2ma(list=&p, ma=_p);
   %let cticker = &_p1;
   %let risklev = &_p2;
   %let _p0 = %eval(&_p0 - 2);

   %let maxfa = &_p0;

   %do i = 0 %to &maxfa;
      %global stkwgt&i stktck&i;
   %end;
   %let stkwgt0=&maxfa;
   %let stktck0=&maxfa;

   proc sql noprint;
   select pfid,pflname
   %do i = 1 %to &_p0;
      ,t&i, put(100*w&i,z2.)
   %end;
   into :pfid, :pflname
   %do i = 1 %to &_p0;
      , :stktck&i, :stkwgt&i
   %end;
   from product.pfmaster
      where pfname = "&pfname";

   %let sumstklev = 0;

   %do i = 1 %to &stkwgt0;
      %let sumstklev = %eval(&sumstklev + &&stkwgt&i);
   %end;

%mend;
```

optimize_pf.macro


```
%macro opto_pf(pfid=);
/* optimize a portfolio
input:
product.plast(cticker,ticker) plan - assets
product.opop(lambda,kappa,linear,minval) optimizer run options
product.pfmaster(pfid,cticker,resklev,fa1,...fan) - portfolio list with fixed asset le
vels
product.fppdtt(date,ticker1,...) return series for all assets&indexes used in this run
product.asset_tv(ticker,....) asset characteristics for all assets used in this run
pod!pop2_ini(RUNID,PFNAME,ticker,wgt) inital values
pod!pop2_aaconst(RUNID,PFNAME,var,target,range) asset allocation constraints
pod!pop2_tyconst(RUNID,PFNAME,typevar,typeval) asset type constraints
output:
pf.out_ctw(cticker,ticker,wgt) - model output portfolio
*/

   %put Optimizing Portfolio &pfid &pflname;

   %ds2macro(in_ds=product.opop); /* get optimizer options */


   proc sql;
   create table pf.plast as
      select "&pfid" as cticker, ticker, fname
      from product.plast
      where cticker = "&cticker"
      order by ticker;

   %global need2run;

   %let need2run = 1;

   %special_cases;


   %if &need2run eq 1 %then %do;

      %getopini(RUNID=&RUNID, PFNAME=&PFNAME, out_ctw=pf.initds);  /* get initial valu
es for optimizer */

      %getaacon(RUNID=&RUNID, PFNAME=&PFNAME, cticker=&cticker, risklev=&risklev, aaco
nst=pf.aaconst); /* get asset allocation constraints */

      %gettycon(RUNID=&RUNID, PFNAME=&PFNAME, tyconst=pf.tyconst); /* get asset type c
onstraints */

      %getfacon(faconst=faconst); /* get fixed asset constraints */

      data sconst;
      merge pf.tyconst faconst;
      by ticker;
      if wgt ne . then do;
         maxwgt=wgt;
         minwgt=wgt;
         end;
      run;

      %cleanret(in_tv=pf.plast,in_dtt=product.fpp_dtt,out_dtt=pf.mdl_dtt);


      %ret2tv(in_tv=pf.plast, in_dtt=pf.mdl_dtt);

      %taradj(inconst=pf.aaconst, outconst=pf.adjconst, in_dtt=product.fpp_dtt, fixed=
faconst , in_tv= product.asset_tv);

      %if &status eq 0 %then %do;
         %mk_bendtt(in_tv=pf.plast,in_dtt=product.fpp_dtt,in_targ=pf.adjconst, out_dtt
=pf.mdl_dtt);

         %opto_ctw(in_ctw=pf.initds, in_dtt=pf.mdl_dtt, bench=_BENCH, in_tv=product.as
set_tv,
                   lambda=&lambda, kappa=&kappa, linear=&linear, tconst=pf.adjconst, s
const=sconst,
                   out_ctw=pf.ctw, minval=&minval,inquad=pf.inquad);

         %if &status eq 0 %then %do;
            %ctw2whole_pct(in_ctw=pf.ctw, in_dtt=pf.mdl_dtt, out_ctw=pf.mdl_ctw);
            %terr_ctw(in_ctw=pf.mdl_ctw,bench=_BENCH,in_dtt=pf.mdl_dtt,out_ds=pf.mdl_t
err,peryear=12,out_dtt=,rebal=0);
            %put Optimization succeeded;
            %dbexec(db=POD, sql=%bquote(pop2q_setstatus '&RUNID', '&PFNAME', 'done'));
         %end;
         %else %do;
            %put Optimization failed;
            %dbexec(db=POD, sql=%bquote(pop2q_setstatus '&RUNID', '&PFNAME', 'fail'));
         %end;
      %end;
      %else %do;
         %put New target not possible. Optimization abandoned;
         %dbexec(db=POD, sql=%bquote(pop2q_setstatus '&RUNID', '&PFNAME', 'notappl'));
      %end;
   %end;
%mend;

%macro setstatus(RUNID=, PFNAME=, status= );

  %dbexec(db=POD, sql=%bquote(pop2q_setstatus '&RUNID', '&PFNAME', '&status'));

%mend;


%macro getopini(RUNID=, PFNAME=, out_ctw=pf.initds);
/* get optimizer initial values
*/
   %dbgetpfx(db=pod,ds=initds)
      select ticker, wgt
      from pop2_ini
      where RUNID=%bquote('&RUNID')
      and PFNAME = %bquote('&PFNAME')
      order by ticker
      );
   %if %sysfunc(exist(initds)) %then %do;
      data &out_ctw;
      merge pf.plast initds;
      by ticker;
      if wgt eq . then wgt = 0;
      run;
   %end;
   %else %do;
      proc sql noprint;
      select count(*) into :astcount
         from pf.plast;

      data &out_ctw;
      set pf.plast;
      wgt=1/&astcount;
      run;
   %end;

%mend;
```

```
%macro getaacon(RUNID=, PFNAME=, cticker=, risklev=, aaconst=);
/* get asset allocation constraints
*/


   %dbgetpfx(db=POD,ds=&aaconst)
      select 1 as priority, var,target,range
      from pop2_aaconst
      where RUNID=%bquote('&RUNID')
      and PFNAME=%bquote('&PFNAME')
      union
      select 2 as priority, var,target,range
      from pop2_aaconst
      where RUNID=%bquote('&RUNID')
      and PFNAME=%bquote('&cticker/&risklev')
      union
      select 3 as priority, var,target,range
      from pop2_aaconst
      where RUNID='DEFAULT'
      and PFNAME=%bquote('DEFAULT/&risklev')
      order by var, priority
      );

   data &aaconst(drop=irange);
   set &aaconst(rename=(range=irange));
   retain range;
   by var priority;
   if first.var then range = irange;
   minwgt = target - range;
   if minwgt < 0 then minwgt = 0;
   maxwgt = target + range;
   if maxwgt > 1 then maxwgt = 1;
   if last.var then output;
   run;

%mend;


%macro gettycon(RUNID=, PFNAME=, tyconst=);
/* get asset type constraints
*/

   %dbgetpfx(db=pod,ds=_const)
      select 1 as priority, typevar,typeval,minwgt,maxwgt
      from pop2_tyconst
      where RUNID=%bquote('&RUNID')
      and PFNAME=%bquote('&PFNAME')
      union
      select 2 as priority, typevar,typeval,minwgt,maxwgt
      from pop2_tyconst
      where RUNID=%bquote('&RUNID')
      and PFNAME=%bquote('&cticker/&risklev')
      union
      select 3 as priority, typevar,typeval,minwgt,maxwgt
      from pop2_tyconst
      where RUNID=%bquote('&RUNID')
      and PFNAME=%bquote('')
      order by typevar,typeval,priority
      );
   data _const;
   set _const;
   by typevar typeval priority;
   if first.priority;
   run;

   %ds2ma(in_ds=_const, var=typevar, ma=_tvr, uniq=0, num=0);
   %ds2ma(in_ds=_const, var=typeval, ma=_tvl, uniq=0, num=0);
   %ds2ma(in_ds=_const, var=minwgt,  ma=_tmn, uniq=0, num=1);
   %ds2ma(in_ds=_const, var=maxwgt,  ma=_tmx, uniq=0, num=1);

   proc sql noprint;
   create table &tyconst(ticker char(8), minwgt num, maxwgt num);

   %local i;
   %do i = 1 %to &_tvr0;
      proc sql noprint;
      create table _const as
         select a.ticker,&&_tmn&i as minwgt, &&_tmx&i as maxwgt
         from pf.plast a, pop2mast.asset_tv b
         where a.ticker=b.ticker
         and b.&&_tvr&i = "&&_tvl&i";

      %dsappend(i=&i,in_ds=_const,out_ds=&tyconst);
   %end;
   proc sort data=&tyconst;
   by ticker;
   run;

   %ty_special_cases;


%mend;

%macro getfacon(faconst=);
/* get fixed asset constraints ticker,minwgt,maxwgt
*/

   %local i;

   data &faconst;
   format ticker $8. wgt best.;
   if &stktck0 > 0;
   %do i = 1 %to &stktck0;
      ticker = "&&stktck&i";
      wgt = &&stkwgt&i/100;
      minwgt=wgt;
      maxwgt=wgt;
      if ticker ne '' then output;
   %end;
   run;

%mend;


%macro mk_bendtt(in_tv=,in_dtt=,in_targ=,out_dtt=);
/* make benchmarks for the portfolios benchi is benchmark for ipq i */

   %ds2l(in_ds=&in_tv, var=ticker, list=plops );

   proc transpose data=&in_targ(keep=var target) out=fpp_targ(drop=_name_);
   id var;
   var target;
   run;

   data fpp_targ(rename=(stocks=stocks_ fstocks=fstocks_ bonds=bonds_ cash=cash_ hybon
ds=hybonds_));
   format stocks  fstocks bonds hybonds  cash 8.4;
   set fpp_targ;
```

```
    run;

    data &out_dtt(keep=date _bench &plops);
    merge fpp_targ &in_dtt;
    retain Tstocks_  Tfstocks_ Tbonds_ Thybonds_ Tcash_;
    array _ac(*) stocks_ fstocks_ bonds_ hybonds_ cash_;
    array _tac(*) Tstocks_  Tfstocks_ Tbonds_ Thybonds_ Tcash_;
    if _n_ eq 1 then do;
       do i = 1 to 5;
          if _ac[i] eq . then _ac[i] = 0;
          _tac[i] = _ac[i];
       end;
    end;
    _bench = (tstocks_-tfstocks_)*_w5000 + tfstocks_*_eafe + (tbonds_-thybonds_)*_bond
+thybonds_*_ml_hym + tcash_*_cash_ ;
    run;

%mend;
```

```
%macro pre_cons;
/* pre-optimization consolidate and subset required input data
*/

   /* set up the plan assets: map proxies, remove closed,exempt, unchanged */

   data product.proxymap(keep=cticker ticker fprs_id fname);
   set product.rawproxy(where=(ticker ne '' or xfprs_id ne hard_id) rename=(xf_fname=f
name plan=cticker));
   format fprs_id $4. ;
   fprs_id = substr(xfprs_id,2,4);
   ticker = hard_id;
   run;

   %let mo = %substr(&refdate,5,2);

   %if %eval((&mo / 3) * 3 eq &mo) %then %let sflag = 'X';
   %else %let sflag = 'U';


   proc sql;
   create table product.plans as
      select distinct plan, plann as planname, c_id as sponid, c_name as sponname, seg
ment, status
      from product.rawplast
      where status ne &sflag
      order by plan;

   proc sort data=product.proxymap;
   by cticker fprs_id;
   proc sort data=product.rawplast;
   by plan fprs_id;
   run;

   data product.plast(keep=cticker ticker fname fprs_id isproxy costk);
   merge product.rawplast(in=inplast rename=(plan=cticker f_name=fname)) product.proxy
map(in=inproxy) ;
   by cticker fprs_id;
   if inplast and ((closed ne 'Y' and exempt ne 'Y') or cticker = 'FPP99998') and stat
us ne &sflag;
   if inproxy then isproxy = 'Y';
   else isproxy = 'N';
   if ticker eq '' then ticker = '_'|| fprs_id;
   run;

   proc sort data=product.plast;
   by cticker ticker;
   run;

   data product.plast;
   set product.plast;
   by cticker ticker;
   if first.ticker;
   run;


   proc sql;
   create table livetick as
      select distinct ticker from product.plast
      order by ticker;

   data a;
   length ticker $8;
   ticker = 'VTSMX';
   run;

   proc append base=livetick data=a;
   proc sort data=livetick;
   by ticker;
   run;


   /* build asset_tv */
   %mk_asset_tv(in_t=livetick,use_mstar=1,use_fame=1,use_fprs=1);

   /* now get returns */


   %dt_plusm(in_date=&refdate, months=-35, out_date=retfrom, dd=0);

   %mk_dtt(from_dt=&retfrom, to_dt=&refdate,asset_tv=product.asset_tv,stock=1,fund=1,o
ther_dtt=product.fprs_dtt);

   /* chuck out those without all the returns */

   proc sql;
   delete from product.plast
      where ticker in (select ticker from product.missret) and costk eq 'N' and cticke
r ne 'FPP99998';

   /* chuck out those with bad objectives */

   %objfilter(in_ds=product.asset_tv,out_ds=ok_tv);

   proc sql;
   create table product.plast as
      select * from product.plast
      where ticker in (select ticker from ok_tv) or cticker eq 'FPP99998'
      order by cticker,ticker;


   %mkproxy(in_ds=product.plast, keyvar=CTICKER, proxyvar=proxyplan, sigvars=TICKER, m
ap_ds=product.plast);

   proc sql;
   create table cp as
   select distinct cticker as plan,proxyplan
      from product.plast
      order by plan;

   data product.plans;
   merge product.plans cp;
   by plan;
   if proxyplan eq '' then proxyplan = plan;
   run;


   %set_aaconst;

%mend;

%macro set_aaconst;
/*

   %let x = fppgrp401kcvt "&RUNID", "&PRODDATE";

   %dbexec(db=pod,sql=%bquote(&x));
*/

   /* adjust conservative constraints for GIC and no cash case */
```

```
   %dblibname(db=POD);

   proc sql;
   create table hasconst as
      select PFNAME
      from pod.pop2_aaconst
      where RUNID="&RUNID"
      order by PFNAME;

   proc sql;
   create table giconly as
   select distinct compress(cticker||'/5') as PFNAME from product.plast a
      where ticker = '_GIC_'
      and cticker not in (select cticker from product.plast where ticker = '_CASH_')
      union
   select distinct compress(cticker||'/4') as PFNAME from product.plast a
      where ticker = '_GIC_'
      and cticker not in (select cticker from product.plast where ticker = '_CASH_')
;
   create table giconly as
      select * from giconly
      where PFNAME not in (select PFNAME from hasconst);


     proc sql;
     create table _dfo as
     select "&runid" as runid, b.PFNAME,a.var,a.target,a.range
      from pod.pop2_aaconst a, giconly b
      where a.runid="fpp/PRODUCTION/DEFAULT"
      and a.PFNAME="DEFAULT/5"
      and var in ('STOCKS','FSTOCKS','OTHER', 'HYBONDS');

   create table _bc as
      select "&runid" as runid, b.PFNAME,a.var,a.target,.15 as range
      from pod.pop2_aaconst a, giconly b
      where a.runid="fpp/PRODUCTION/DEFAULT"
      and a.PFNAME="DEFAULT/5"
      and var in ('BONDS', 'CASH');

   proc append base=pod.pop2_aaconst(bulkcopy=yes) data=_dfo;
   run;

   proc append base=pod.pop2_aaconst(bulkcopy=yes) data=_bc;
   run;

%mend;
```

```
%macro special_cases;


   /* for "cash" risk level, use money market fund if there, otherwise use GIC, otherw
ise optimize */

   %if &risklev eq 6 %then %do;

      %if %eval(&sumstklev > 0) %then %do;
         %dbexec(db=POD, sql=%bquote(pop2q_setstatus '&RUNID', '&PFNAME', 'notappl'));
         %let need2run = 0;
      %end;
      %else %do;

         %let hascash = 0;
         proc sql;
         select 1 into :hascash
            from pf.plast where ticker= '_CASH_';

         %if &hascash eq 1 %then %do;
            %mkctw(cticker=&pfid, in_tv=, tickers=_CASH_, tick_ma=, wgt=1,wgts=, wgt_m
a=, out_ctw=pf.mdl_ctw);
            %dbexec(db=POD, sql=%bquote(pop2q_setstatus '&RUNID', '&PFNAME', 'done'));
            %let need2run = 0;
         %end;
         %else %do;

            proc sql;
            select 1 into :hascash
               from pf.plast where ticker= '_GIC_';
            %if &hascash eq 1 %then %do;
               %mkctw(cticker=&pfid, in_tv=, tickers=_GIC_, tick_ma=, wgt=1,wgts=, wgt
_ma=, out_ctw=pf.mdl_ctw);
               %dbexec(db=POD, sql=%bquote(pop2q_setstatus '&RUNID', '&PFNAME', 'done'
));
               %let need2run = 0;
            %end;
         %end;
         data pf.adjconst;
         set pf.aaconst;
         run;
      %end;

   %end;

   /* for 99998 case (asset allocation fund solution) hard map the solutions to specif
ic assets */

   %if &cticker eq FPP99998 %then %do;
      %if &risklev eq 1 or &risklev eq 2 %then %let ticker = FAMRX;
      %if &risklev eq 3                  %then %let ticker = FASGX;
      %if &risklev eq 4 or &risklev eq 5 or &risklev eq 6 %then %let ticker = FASIX;
      %mkctw(cticker=&pfid, in_tv=, tickers=&ticker, tick_ma=, wgt=1,wgts=, wgt_ma=, o
ut_ctw=pf.mdl_ctw);
      %dbexec(db=POD, sql=%bquote(pop2q_setstatus '&RUNID', '&PFNAME', 'done'));
      data pf.adjconst;
      set pf.aaconst;
      run;
      %let need2run = 0;
   %end;


%mend;

%macro ty_special_cases;


   /* if we don't use kappa, and it is all equity then if there is a shortage of equit
y funds,
      relax the 25% max to 35%
   */

   %if %eval(&kappa eq 0 and &risklev eq 1) %then %do;

      proc sql;
      select sum(.25 * stocks)
         into :stockmax
         from pf.plast a, product.asset_tv b
         where a.ticker = b.ticker;

      %if %eval(&stockmax < 1) %then %do;

         data pf.tyconst;
         set pf.tyconst;
         if maxwgt eq .25 then maxwgt = .35;
         run;

      %end;

   %end;

   %if %eval(&risklev eq 1 and &sumstklev eq 0) %then %do;
      proc sql noprint;
      create table alleqconst as
         select a.ticker format $8.
         from pf.plast a, product.asset_tv b
         where a.ticker=b.ticker
         and b.atype not in ('EQUITY','FOREIGN')
         order by ticker;
      data pf.tyconst(keep=ticker minwgt maxwgt);
      merge pf.tyconst(in=int) alleqconst(in=ina);
      by ticker;
      if ina then do;
         minwgt=0;
         maxwgt=0;
      end;
      run;
   %end;

   %local gicmin;
   %let gicmin = 0;
   %if (&risklev eq 3) %then %let gicmin = .05;
   %if (&risklev eq 4) %then %let gicmin = .25;
   %if (&risklev eq 5) %then %let gicmin = .5;

   data gicon;
   length ticker $8.;
   ticker = '_GIC_';
   minwgt = &gicmin;
   maxwgt = 1;
   output;
   ticker = '_0632';
   output;
   ticker = '_0633';
   output;
   run;
   proc append base=pf.tyconst data=gicon;
   proc sort data=pf.tyconst;
   by ticker;
   run;
```

```
%mend;

%macro special_constraints;
/*
   data pf.adjconst;
   set pf.adjconst end=eof;
   output;
   if eof then do;
      var='HYBONDS';
      target = 0;
      range = .1;
      minwgt=0;
      maxwgt = .1;
      output;
   end;
   run;

   proc sort data=pf.adjconst;
   by var;
   run;
*/

%mend;
```

```
%macro ctw2whole_pct(in_ctw=, in_dtt=, out_ctw=);

   data finals;
   set &in_ctw(drop=cticker);
   wgt=round(wgt,.0001);
   run;
   proc sort data=finals;
   by ticker;

   data aret;
   set &in_dtt;
   run;

   %dtt2dtr(in_dtt=&in_dtt,out_dtr=_c2dtr);

   proc sql noprint;
   create table preset as
   select a.ticker,a.wgt*100 as wgt, std(return) as std
      from finals a, _c2dtr b
      where a.ticker=b.ticker
     group by a.ticker,wgt
      order by ticker;

   data preset;
   set preset;
        if ( wgt ne int(wgt) ) then do;
            lowerbd = int(wgt);
            upperbd = int(wgt) + 1  ;
        end;
        else do;
            lowerbd = wgt;
            upperbd = wgt;
        end;
        integer=1;
        wstd= std*wgt;
   run;

   proc means data=preSet noprint;
        var wstd;
        output out=_xsum sum=std;

   data _xsum;
        length ticker $10;
        set _xsum;
        ticker = "SUM";
        wgt = -1;
        lowerbd=-1;
        upperbd=-1;
        integer=1;
        wstd=.;

   data preset;
   set preset _xsum;
   run;

   proc transpose data=preSet out=inSetx;
        id ticker;

   data inSet;
        length _type_ $15;
        set insetX(rename=(_name_=_id_) );

        if ( _id_ in ("LOWERBD","UPPERBD") ) then do;
            _type_ = _id_;
            _rhs_ = .;
            output;
        end;
        else if ( _id_ in ("STD") ) then do;

            _type_ = "MIN"; _rhs_=.; output;
            _type_ = "GE" ; _rhs_=0; output;
        end;
        else if ( _id_ in ("INTEGER") ) then do;
            _type_=_id_ ; _rhs_=.; output;
            SUM=0;_type_="EQ";_rhs_=100; output;
        end;

   proc lp data=inSet printlevel=-2 primalout=_lpout activeout=actout dualout=duout
      imaxit=100000 time=25 varselect=far
      iepsilon=.001 canselect=lifo maxit1=200;
   run;

   proc sql noprint;
   create table &out_ctw as
      select a.cticker length=8,a.ticker length=8,b._VALUE_/100 as wgt
      from &in_ctw a, _lpout b
      where a.ticker=b._VAR_
      and b._TYPE_ in ("FIXED","INTEGER")
      and b._VALUE_ ne 0
      order by a.cticker,a.ticker;


%mend;
```

```
/* discrete optimizer
*/

%macro opto_ctw(in_ctw=, in_tv= , in_dtt=, bench=, lambda=1, kappa=0, linear=alpha, tc
onst=, sconst=, gconst=, out_ctw=, minval=, trunc=,inquad=inquad,outquad=outquad,qopti
ons=,covmat=,maxiter=50);
/* perform optimization on many portfolios
  in_ctw(cticker,ticker,wgt) list of portfolios with initial weights of assets to opti
mize
  in_dtt(date,ticker,ticker,...) return series used for variance calculation
  bench - ticker of benchmark
  lambda - covariance multiplier. defaults to 1.
  kappa - concentration multiplier. defaults to 0.
  linear - name of var to get linear term of objective function from. optional
  in_tv(ticker,&linear,...) - values of linear values. required if &linear is present
  initval(ticker,wgt) - asset-weight ds of initial values. optional
  tconst(var,minwgt,maxwgt) - total constraints dataset. optional
  sconst(ticker,minwgt,maxwgt) - specific constraints dataset. optional
  minval - the minimum nonzero weight of any asset. defaults to zero
  trunc - if nonempty, truncate to zero those ticker values less than &trunc
  out_ctw(cticker,ticker,wgt) recommended portfolios
 covmat - optional covariance matrix
*/
/* get covariance matrix */

   %local i p gotp;
   %global status ziter;
   %let ziter = 0;
   %let gotp = 0;

   %if &in_ctw eq %then %do;
      data _in_ctw;
      length cticker $8;
      retain cticker 'CTW';
      set &in_tv(keep=ticker);
      wgt = 1;
      run;
      %let in_ctw=_in_ctw;
   %end;

   %ds2ma(in_ds=&in_ctw,var=ticker,ma=_ma);
   %ds2ma(in_ds=&in_ctw,var=cticker,ma=_p);
   proc sql noprint;
   create table _ticks as
      select distinct ticker from &in_ctw;
   insert into _ticks(ticker) values ("&bench");
   proc sort data=_ticks;
   by ticker;
   run;
   %if &covmat eq %then %do;

      %cleanret(in_tv=_ticks,in_dtt=&in_dtt,out_dtt=_dtt2);

      %if &bench ne %then %do;

         data _dtt1(drop=_name);
         set _dtt2;
         format _name $8.;
         array __x _numeric_;
         do over __x;
            call vname(__x,_name);
            if _name ne "&bench" and _name ne "DATE" then __x = __x - &bench;
            end;
      %end;

      proc corr cov data=_dtt1 outp=_covmat nosimple noprint;
      var %do i = 1 %to &_ma0; &&_ma&i %end;;
      run;
      %let covmat = _covmat;

   %end;

   %ctwnorm(in_ctw=&in_ctw,out_ctw=_ctw);


   %do p = 1 %to &_p0;

      %put Optimizing &p / &_p0 : &&_p&p lambda=&lambda kappa=&kappa linear=&linear mi
nval=&minval;

      data _initval(keep=ticker wgt);
      set _ctw(where=(cticker = "&&_p&p"));
      run;

      proc sort data=_initval;
      by ticker;
      run;

      %if &sconst.&gconst eq %then %do;
         data _nullsc;
         format ticker $8.;
         set _initval(keep=ticker);
         retain maxwgt 1 minwgt 0;
         run;
         %let sconst = _nullsc;
      %end;

      data _in_tv;
      merge _initval(keep=ticker in=ina) &in_tv;
      by ticker;
      if ina;
      run;

      %if %sysevalf(&minval > 0) %then %do;
         %discopt(in_tv=_in_tv, covmat=&covmat, lambda=&lambda, kappa=&kappa, linear=&
linear, initval=_initval,
                  tconst=&tconst, sconst=&sconst,qoptions=&qoptions,outquad=&outquad,m
inval=&minval,trunc=&trunc,inquad=&inquad,gconst=&gconst,maxiter=&maxiter);
      %end;
      %else %do;
         %quad(in_tv=_in_tv, covmat=&covmat, lambda=&lambda, kappa=&kappa, linear=&lin
ear, initval=_initval,
               tconst=&tconst, qoptions=&qoptions,sconst=&sconst, outquad=&outquad,inq
uad=&inquad,gconst=&gconst);
      %end;

      %if "&objfnval" ne "." %then %do;
         %let status = 0;
         %let gotp = %eval(&gotp + 1);

         %q2ctw(out_ctw=_out_ctw,cticker=&&_p&p,outquad=&outquad);
         %if &p eq 1 %then %do;
            data &out_ctw;
            length cticker ticker $8;
            set _out_ctw;
            run;
         %end;
         %else %do;
            proc append base=&out_ctw data=_out_ctw;
            run;
```

```
        %end;
      %end;
      %else %let status = 1;

   %end;


%mend;

%macro opto_res(mdl_ctw=,in_dtt=,bench=,out_ds=,trackerr=0,cumret=0);
/* consolidate optimization results
mdl_ctw(cticker,ticker,return) - optimization output model
in_dtt(return series for use in tracking error and/or cumret calculations
bench - benchmark for tracking error calculation
out_ds(ticker/stat, cticker1,cticker2,...) model results transposed, with statistics l
ines
trackerr - nonzero to indicate tracking error should be calculated
cumret - nonzero to indicate cumulative return should be calculated
*/

   proc sql noprint;
   select count(*)
      into :results
      from &mdl_ctw;
   %if %eval(&results ne 0) %then %do;

      %l2wtran(in_ds=&mdl_ctw,out_ds=&out_ds,sort=1,var=wgt,by=ticker,id=cticker);

      %if &trackerr ne 0 %then %do;
         %terr_ctw(in_ctw=&mdl_ctw,in_dtt=&in_dtt,bench=&bench,out_ds=_terr1,peryear=1
2);

         proc transpose data=_terr1(rename=(trackerr=__track)) out=_terr1(rename=(_nam
e_=ticker));
         id ticker;
         var __track;
         run;

         proc append base=&out_ds data=_terr1 force;
         run;
      %end;

      %if &cumret ne 0 %then %do;
         %cleanret(in_tv=&mdl_ctw,in_dtt=&in_dtt,out_dtt=_dtt);
         %totalret(in_dtt=_dtt,out_dtt=_dtt);
         %ctwret(in_ctw=&mdl_ctw,in_dtt=_dtt,out_dtt=_dtt1);
         data _dtt1;
         set _dtt1(drop=date);
         format ticker $8.;
         retain ticker '__RETURN';
         run;
         proc append base=&out_ds data=_dtt1 force;
         run;
      %end;
      proc sort data=&out_ds;
      by ticker;
      run;
   %end;
   %else %do;
      proc sql noprint;
      create table &out_ds (ticker char(8));
   %end;

%mend;
```

```
%macro forcemin(in_ctw=, const=, minval=.05);

   proc sql;
   create table _force as
      select a.ticker,a.wgt,b.lower,b.upper
      from &in_ctw a left join &const b on a.ticker=b.ticker
      order by a.wgt;

   data &const;
   set _force nobs=_nobs;
   if &minval*(_nobs-_n_) > 1 or wgt < &minval/2 then do;
      upper = 0;
      lower = 0;
   end;
   else do;
      if lower < &minval then lower = &minval;
      if upper eq . then upper = 1;
   end;
   run;

%mend;


%macro dist2min(in_ctw, minval=.05);

   %global dist2min;

   data _null_;
   retain distance 0;
   set &in_ctw end=eof;
   if wgt < &minval then do;
      if wgt < &minval/2 then distance = distance + wgt;
      else distance = distance + &minval - wgt;
   end;
   if eof then call symput("DIST2MIN",put(distance,8.6));
   run;

%mend;


%macro discopt(in_tv=, covmat=, lambda=1, kappa=0, linear=, initval=, tconst=, sconst=
,gconst=, outquad=,minval=.05,trunc=,inquad=inquad,qoptions=,maxiter=50);
/* discrete optimization
  in_tv(ticker,&linear,...) - asset list of assets to be optimized. required.
  covmat - covariance matrix of assets in in_tv. required.
  kappa - concentration multiplier. defaults to 0.

  lambda - covariance multiplier. defaults to 1.
  linear - name of var to get linear term of objective function from. optional
  initval(ticker,wgt) - asset-weight ds of initial values. optional
  tconst(var,minwgt,maxwgt) - total constraints dataset. optional
  sconst(ticker,minwgt,maxwgt) - specific constraints dataset. optional
  outquad - proc nlp output dataset. optional: if absent proc nlp is not called.
  minval - the minimum nonzero weight of any asset
  trunc - if nonempty, truncate to zero those ticker values less than &trunc
   this is how we optimize subject to any given
   ticker being either = 0 or >= &minval. The idea:
   optimize the unconstained set. if the optimimum happens to be in the
   constained subset, we're done.  Otherwise pick any ticker where
   0 < wgt < &minval
   and optimize twice: once with that ticker = 0 and once with it >= &minval.
   for each of those optimizations, if it's worse than the best we've
   done so far, don't continue down that path.  If it's better, recurse.
*/
```

```
/*  options mprint symbolgen notes;  */
   %global stklen olddist dist2min;
   %global status;
   %global objfnval;
   %global quadcall quadlast;
   %let olddist = 1;
   %let quadcall = 0;
   %let quadlast = 0;
   %let dist2min = 999;


   %initstk;
   %ds2ma(in_ds=&in_tv,var=ticker,ma=_qt);
   /* set up inquad.  Only 'upper' and 'lower' are changed */

   %quad(in_tv=&in_tv, covmat=&covmat, lambda=&lambda, kappa=&kappa, linear=&linear, i
nitval=&initval, tconst=&tconst, sconst=&sconst,inquad=&inquad,qoptions=&qoptions,gcon
st=&gconst);
   proc transpose data=&inquad(drop= where=(_name_="LOWER")) out=l(rename=(_name_=tick
er));
   proc transpose data=&inquad(drop= where=(_name_="UPPER")) out=u(rename=(_name_=tick
er));
   proc sort data=u;
   by ticker;
   proc sort data=l;
   by ticker;
   run;

   data constrai;
   merge u l;
   by ticker;
   prev = -1;
   run;

   %pushone(constrai);

   %let minobjfn = 99999;
   %let iter = 0;
   %let ranforce = 0;
   %let potentl = .;
   %let objfnval = ;

   %do %while(&stklen > 0);
      %let iter = %eval(&iter + 1);
      %popstk(constrai,&minval);
      %if (&popped) %then %do;
         data _null_;
         title "Iteration &iter minobjfn = &minobjfn last objfnval = &objfnval ";
         run;
         %runcopt(minval=&minval,inquad=&inquad,qoptions=&qoptions);
         %if &feasible & %sysevalf(&minobjfn > &objfnval ) %then %do;
            %if &hitick eq '' %then %do;
               %put &stklev.************************** new low = &objfnval vs potent
ial &potentl on quadcall &quadcall markx*******;
               %let minobjfn = &objfnval;
               %let quadlast = &quadcall;
               data &outquad;
               set toutquad;
               run;
            %end;
            %else %do;
               %if &trunc ne %then %trimlow(constrai,&trunc,&minval);
               %pushstk(constrai,&hitick,&maxlf,&lowtick,&mingz,&minval);
               %if &ranforce ne 1 %then %do;
                  %let potentl = &minobjfn;
                  proc sql;
                  select min(objval)
                     into :potentl
                     from stacko;
                  %let potentl = &potentl;
                  /* try to force away stuff that we zero on the first round */
/*
                  %popstk(constrai,&minval);
                  proc sql;
                  create table _nc as
                     select a.ticker,a.wgt,b.lower,b.upper
                     from _qctw a left join constrai b on a.ticker=b.ticker
                     order by a.wgt;
                  data constrai(drop=wgt);
                  set _nc;
                  if wgt = 0 then do;
                     upper = 0;
                     lower = 0;
                  end;
                  run;
                  %pushstk(constrai,&hitick,&maxlf,&lowtick,&mingz,&minval);
*/
                  /* end of that attempt */
                  %forcemin(in_ctw=_qctw, const=constrai, minval=&minval);
                  proc print data=_force;
                  title "_force";
                  proc print data=constrai;
                  title "constrai";
                  run;
                  %runcopt(minval=&minval,inquad=&inquad,qoptions=&qoptions);
                  %let ranforce = 1;
                  %if &feasible %then %do;
                     %put &stklev.************************** force low = &objfnval v
s potential &potentl on quadcall &quadcall markx*******;
                     %let minobjfn = &objfnval;
                     %let quadlast = &quadcall;
                     data &outquad;
                     set toutquad;
                     run;
                  %end;
               %end;
            %end;
         %end;
      %end;


      %if %sysevalf(&iter ge &maxiter) %then %do;

         %if %sysevalf(&minobjfn ne 99999) %then %do;
           %put quit after &quadcall calls to quad, obj=&minobjfn vs potential &potent
l markx;
         %end;
         %let stklen = 0;
      %end;
   %end;
   %if %sysevalf(&minobjfn ne 99999) %then %do;
      data toutquad;
      set &outquad;
      run;
      %let objfnval = &minobjfn;
      %put Finished after &quadcall calls to quad, obj=&minobjfn markx;
      %let status = SUCCESSFUL;
   %end;
%mend;
```

```
%macro initstk();
/* initialize stack
stack:  each set of records having the same pos go together.
there is one record per ticker.
 pos - position in stack.
 ticker - ticker
 lower - lower bound for that ticker
 upper - upper bound for that ticker
 prev - for just split ticker, this is the wgt in the parent's run
stacko: one record per pos.
 pos - position in stack.
 ticker - ticker that was just split
 cur - lower bound for that ticker (either zero or minwgt)
 prev -  wgt in the parent's run
 objval - weight of parent's run
*/
    %global stklen;
    %global stklev ;
    %let stklev = .;
    %let stklen = 0;
   proc sql noprint;
   create table stack
      (pos num,
       ticker char(8),
       lower num,
       upper num,
       prev num);

   create table stacko
      (pos num,
       ticker char(8),
       objval num,
       cur num,
       prev num,
       opt num,
       dist2min num);

%mend;


%macro popstk(constrai,minval);
/* pop constraint set from stack
*/

  %global stklen;
  %global stklev;
  %global popped;
  %let sobjval = %sysevalf(&minobjfn+1);

  %do %while (%sysevalf(&sobjval > &minobjfn) & %eval(&stklen > 0));
     %if &stklev eq . %then %let stklev = ; %else %if %eval(&stklen <= 100) %then %let
 stklev = %substr(&stklev,2);

     proc sql noprint;
     create table _conpos
        as select so.objval, so.dist2min, so.opt, so.pos, count(*) as cnt
        from stack s, stacko so
        where s.pos=so.pos
        and s.upper ne 0
        and s.lower ne &minval
        group by so.dist2min, so.objval, so.opt, so.pos
        order by so.dist2min, so.objval, so.opt, so.pos;
/*
        order by so.pos;
*/
     %let cntcnt = 0;
     proc sql;
     select count(*)
        into :cntcnt
        from _conpos;
     %let cntcnt = &cntcnt;

     %let minpos=0;

     data _null_;
     set _conpos(obs=1);
     call symput("minpos",trim(left(put(pos,8.))));
     run;

     data &constrai stack;
     set stack;
     if pos = &minpos then do;
        output &constrai;
     end;
     else do;
        if pos > &minpos then pos = pos - 1;
        output stack;
     end;
     run;

     data stacko;
     set stacko;
     if pos = &minpos then do;
        call symput("sobjval",objval);
        put "&stklev.popping " ticker ":(" objval "at " prev ") -> " cur "out of &cntc
nt" ;
     end;
     else do;
        if pos > &minpos then pos = pos - 1;
        output;
     end;
     run;

     %let stklen = %eval(&stklen - 1);
  %end;
  %if %sysevalf(&sobjval > &minobjfn) %then %do; %let popped = 0; %end;
  %else %do; %let popped = 1; %end;
%mend;



%macro pushbnd(constrai,ticker,prev,lower,upper);
/* push bounds
*/
   %global stklen;
   %global stklev;


   %let stklen = %eval(&stklen + 1);
   %put &stklev.pushing &ticker :(&objfnval at &prev ) -> &lower .;
   %if %eval(&stklen < 100) %then %do;
      %let stklev = &stklev..;
   %end;
   data _const;
   set &constrai;
   pos = &stklen;
   if ticker = "&ticker" then do;
      lower = &lower;
      upper = &upper;
      prev = &prev;
```

```
    end;
    proc append base=stack data= _const;

    data _consto;
    format ticker $8.;
    pos = &stklen;
    ticker = "&ticker";
    cur = &lower;
    prev = &prev;
    objval = &objfnval;
    opt = abs(prev-cur);
    dist2min = &dist2min;
    run;
    proc append base=stacko data=_consto;

%mend;



%macro pushstk(constrai,tickerh,prevh,tickerl,prevl,minval);
/* push constaints onto stack.
   this pushes 2 sets to stack:
   one with the designated ticker = 0
   and one with it >= 5%
*/
  %global stklen;
  %global stklev;


  %if (%sysevalf(&minval-&prevh < &prevl)) %then %do;
    %pushbnd(&constrai,&tickerh,&prevh,0,0);
    %pushbnd(&constrai,&tickerh,&prevh,&minval,upper);
  %end;
  %else %do;
    %pushbnd(&constrai,&tickerl,&prevl,&minval,upper);
     %pushbnd(&constrai,&tickerl,&prevl,0,0);
  %end;

%mend;


%macro pushone(constrai);
/* push a single item onto stack
*/
  %global stklen;
  %global stklev;
  %let stklen = %eval(&stklen + 1);
  %let stklev = &stklev..;

  data _const;
   length ticker $8;
  set &constrai;
  pos = &stklen;
  prev = -1;

  proc append base=stack data=_const;

  data _consto;
  format ticker $8.;
  pos = &stklen;
  ticker = "";
  cur = -1;
  prev = -1;
   opt = 0;
    objval = -99999;
    dist2min = &dist2min;
   run;
   proc append base=stacko data=_consto;

%mend;



%macro trimlow(constrai,trunc,minval);
/* trimlow
 */
   proc sort data=_qctw;
   by ticker;

   data &constrai(drop=wgt untrim);
   merge &constrai(in=inc) _qctw(drop=cticker in=int) end=eof;
   by ticker;
   retain untrim;
   if int then prev = wgt; else prev=0;
   if _n_ = 1 then untrim = 0;
   if (inc and prev < &trunc and upper ne 0) then do;
       put "&stklev.trimming " ticker " = " prev;
       lower = 0;
       upper = 0;
   end;
   if (upper ne 0 and lower ne &minval) then untrim = untrim + 1;
   if eof then put "&stklev.unconstrained " untrim;
   run;

%mend;

%macro runcopt(minval=,inquad=inquad,qoptions=);
/* set new constraints and run optimization
*/
  %global feasible;
  %global maxlf;
  %global hitick lowtick mingz;
  %let feasible = 0;
  proc transpose data=constrai(keep=ticker lower) out=l;
  id ticker;
  proc transpose data=constrai(keep=ticker upper) out=u;
  id ticker;
  data l;
  set l;
  _name_ = "LOWER";
   _type_ = "lowerbd";

  data u;
  set u;
  _name_ = "UPPER";
   _type_ = "upperbd";
   run;

   data &inquad;
   set &inquad(where=(_name_ not in ("LOWER","UPPER"))) l u;
   run;
   %let quadcall = %eval(&quadcall+1);
   %put &stklev.quad call &quadcall ..;
   %callnlp(ma=_qt,outquad=toutquad,inquad=&inquad,qoptions=&qoptions) ;
   %if ( "&objfnval" ne "." ) %then %do;
      %let feasible = 1;
      %let hitick = ;
      %let maxlf = -1;
```

```
      %q2ctw(outquad=toutquad, out_ctw=_qctw);
      %dist2min(in_ctw=_qctw,minval=&minval);
      proc sql noprint;
      select ticker,wgt
         into :hitick, :maxlf
         from _qctw
         having wgt = (select max(wgt) from _qctw where wgt < &minval - 1E-8 and wgt >
 1E-8);
      select ticker,wgt
         into :lowtick, :mingz
         from _qctw
         having wgt = (select min(wgt) from _qctw where wgt > 1E-8 and wgt < &minval -
 1E-8);
/*       %put ( &hitick ) ( &maxlf ) ( &lowtick) ( &mingz );*/
      %if %sysevalf(&maxlf <= 1E-8) %then %let hitick = '';
   %end;
%mend;
```

```
/* linear factor model
*/

%macro factlin(in_dtr=, in_dtt=, fact_dtt=, factors=, riskfree=_CASH, out_tv=);
/* linear factor model
choose(in_dtt|in_dtr)
in_dtr(date,ticker,return) returns  of assets to model
in_dtt(date,ticker,ticker,...) returns of assets to model
fact_dtt(date,fact1,fact2..) returns of &factors and &riskfree
factors - tickers of factors of model
*/

   %if &in_dtt ne %then %do;
      %let in_dtr = _fl1;
      %dtt2dtr(in_dtt=&in_dtt,out_dtr=&in_dtr);
   %end;

   %if &riskfree ne %then %do;

      data _excdtt(drop=&riskfree _vname);
      merge &in_dtr fact_dtt(keep=date &riskfree);
      by date;
      format _vname $8.;
      array __x _numeric_;
      do over __x;
      call vname(__x,_vname);
         if _vname not in("&riskfree","DATE") then __x = __x - &riskfree;
         end;
      run;
      %let in_dtr = _excdtt;
   %end;


   data _fl2;
   merge &in_dtr &fact_dtt(keep=date &factors);
   by date;
   run;

   proc sort data=_fl2;
   by ticker;
   run;

   options nonotes;

   proc reg data=_fl2 outest=&out_tv(keep = ticker intercep &factors _rmse_ rename=(in
tercep=alpha)) noprint;
   model return = &factors / ADJRSQ MSE ;
        by ticker;
   run;

/*   %trash(_excdtt _fl1 _fl2);*/

%mend;
```

```
%macro alphat(in_tv=, atype=, idx_dtt=, ast_dtr=, factors=, out_mdl=);
/* compute alphas for a given asset type and factors
in_tv(ticker,atype) reference data
atype - asset type to filter &in_tv on
idx_dtt(date,&factors) - factor returns
ast_dtr(date,ticker,return) asset returns
factors - list of factors in regression
out_mdl(ticker,rmse,alpha,_rsq_, _adjrsq_, pvalue, &factors) output model
*/
   %let factcl = %l2cl(list=&factors);
   proc sql noprint;
   select count(*)
      into :_periods
      from &idx_dtt;

   proc sql noprint;
   create table _lretx as
      select a.ticker, a.return-c._cash_ as return, &factcl
      from &ast_dtr a, &in_tv b, &idx_dtt c
      where a.ticker=b.ticker
      and a.date = c.date
      and b.atype = "&atype"
      order by a.ticker;

   select count(*)
      into :cnt
      from _lretx;

   %if %eval(&cnt > 0) %then %do;

      proc reg data=_lretx outest=&out_mdl noprint;
      model return = &factors / ADJRSQ MSE ;
      by ticker;
      run;

      data &out_mdl(keep=ticker rmse alpha &factors _RSQ_ _ADJRSQ_ pvaluem);
      set &out_mdl(rename=(_rmse_=RMSE intercept=ALPHA));
      if rmse eq 0 then alpha = 0;
      if rmse eq 0 then aph=0;
      else aph=(alpha/rmse)*sqrt(&_periods);
      pvaluem = -probt(aph, &_periods-1);
      run;
   %end;
   %else %do;
      data &out_mdl;
      format ticker $8. pvaluem rmse alpha _RSQ_ _ADJRSQ_ best.;
      if 1=2;
      run;
   %end;

%mend;

%macro fppalpha(in_tv=, ast_dtr=, idx_dtt=, mdl_out=);
 /* calculates alphas for all assets
in_tv(ticker,atype,...) assets to compute alphas for. atype indicates which alpha mode
l to use
idx_dtt(date,ticker1,...) return series of indexes
ast_dtr(date,ticker,return) returns series of assets
mdl_out(ticker,alpha,pvalue,...) alpha and pvalue of assets
*/

   data alphadtt;
   set &idx_dtt;
   _LRGSML = _R1000 - _R2000;
   _GROVAL = _R3000G - _R3000V;
   _EXCSTK = _R3000 - _CASH_;
   _EXCBND = _BOND - _CASH_;
        _CREDIT = _ML_HYM - _LB_TR;
        _DURL = _SB_10TR - _SB_TR;
        _DURS = _SB_1YTR - _SB_TR;
        _XMBS = _LB_MBS - _BOND;
        _XFARE = _MS_FARE - _CASH_  ;
        _XEURO = _MS_EURO - _CASH_  ;
        _XAUST = _MS_AUST - _CASH_  ;
        _XNA = _MS_NA - _CASH_  ;
        _XLAM = _IF_LAM - _CASH_  ;
        _XUSD = _USDTW - _CASH_  ;
        _XMUNI =  _LB_MUNI - _CASH_  ;
        _XDURMU = _LB_10MU - _LB_14MU;
   run;

   %alphat(in_tv=&in_tv,atype=EQUITY ,idx_dtt=alphadtt, ast_dtr=&ast_dtr, factors=_LRG
SML _GROVAL _EXCSTK _EXCBND,out_mdl=eqalph) ;
   %alphat(in_tv=&in_tv,atype=BOND   ,idx_dtt=alphadtt, ast_dtr=&ast_dtr, factors=_EXC
BND _CREDIT _DURL _DURS _XMBS,out_mdl=bndalph) ;
/*    %alphat(in_tv=&in_tv,atype=MUNI   ,idx_dtt=alphadtt, ast_dtr=&ast_dtr, factors=_
XMUNI _XDURMU,out_mdl=munialph) ;*/
   %alphat(in_tv=&in_tv,atype=FOREIGN,idx_dtt=alphadtt, ast_dtr=&ast_dtr, factors=_XFA
RE _XEURO _XAUST _XNA _XLAM _XUSD,out_mdl=frnalph) ;

   data &mdl_out;
   set eqalph bndalph /* munialph */ frnalph end=eof;
   if ticker not in ("_CASH_","_GIC_");
   output;
   if eof then do;
      ticker='_CASH_';
      pvaluem=-.5;
      output;
      ticker='_GIC_';
      pvaluem=-.5;
      output;
   end;
   run;

   proc sort data=&mdl_out;
   by ticker;
   run;


%mend;
```

getenv.macro


```
%macro getenv(makenew=0, inpf=0);
/* get environment for fpp run
makenew - zerozero indicates make new product or if inpf nonzero, make new pf director
y
inpf - nonzero indicates we're in a portfolio-specific program, so get the portfolio-s
pecific stuff
*/
   %global RUNID qoptions fppcode dsbase product uproduct prodbase refdate proddate pf
id PFNAME pop2env  pflname output channel feedbase sasoptions;

   %let sasoptions = %sysget(SASOPTIONS);
   options &sasoptions;

   %let RUNID=%sysget(RUNID);
   %let fppcode=%sysget(FPPCODE);
   %let dsbase=%sysget(DSBASE);
   %let feedbase=%sysget(FEEDBASE);
   %let refdate=%sysget(REFDATE);
   %let pop2env = %scan(&runid,2,/);
   %let product = %sysget(PRODUCT);
   %let uproduct = %upcase(&product);

   %let prodbase = &DSBASE/&RUNID;
   %let proddate=%sysget(PRODDATE);

   %let qoptions = %sysget(QOPTIONS);

   %let channel = &feedbase/firsco/&proddate;
   %let output = %sysget(OUTPUT);

   %if &inpf ne 0 %then %do;
      libname product "&prodbase";
      libname baseds "&feedbase/ds/&refdate";
      %let PFNAME=%sysget(PFNAME);
      %pfnam2mac(PFNAME=&PFNAME);
      libname pf "&prodbase/pf/&PFNAME";
      %if &makenew ne 0 %then %trashlib(lib=pf);
    %end;
   %else %do;
      libname product "&prodbase";
      %if &makenew ne 0 %then %trashlib(lib=product);
      libname baseds "&feedbase/ds/&refdate";
   %end;


%mend;
```

mk_pfs.macro


```
%let benindexes = _W5000 _EAFE _CASH_ _BOND _ML_HYM;

%macro setpf(pfn,prod,pd,env);

   %global proddate;

   %if &pd eq %then %let pd = &proddate;
   %else %do;
      %let proddate = &pd;
      x "setenv PRODDATE &proddate";
      x "setenv REFDATE  &proddate";
   %end;


   %if &prod ne %then %do;
      %let proddate=%sysget(PRODDATE);
      %if &env eq %then %let env = PRODUCTION;

      x "setenv PRODUCT &prod";
      x "setenv RUNID &prod/&env/&pd";
   %end;

   %if "&pfn" ne "" %then %do;
      x "setenv PFNAME &PFN";
      %getenv(inpf=1);
   %end;
   %else %getenv;

%mend;

data risklevs;
length rlname $20;
input risklev 1-2 rlname $ 3-22;
cards;
1 Equity Funds Only
2 Aggressive Growth
3 Growth
4 Balanced
5 Conservative
6 Cash Funds Only
;

%macro pfname2desc;

   %global planname rlname stklname ;

   proc sql noprint;
   select planname into :planname
      from product.plans
      where plan = "&cticker";
   select rlname into :rlname
      from risklevs
      where risklev = &risklev;

   %let rlname=&rlname;
   %let planname=%bquote(&planname);
   %if %sysevalf(&sumstklev eq 0) %then %let stklname=;
   %else %let stklname = &sumstklev % Company Stock;

%mend;


%macro mk_pfs(fixedassets=1,risklevs=6);

/* make portfolios
input:
fa_ds
in_portf(cticker,proxyplan) portfolios to optimize
also one for each cticker not in product.fixedast
output:
pf_ds(pfid, pfname, pflname,cticker,risklev,proxyplan,t1..tn,w1..wn) the pfmaster
*/

   %global maxfa;
   %local i;

   %let maxfa = 0;

  proc sql;
   create table product.pfmaster as
      select distinct plan as cticker,proxyplan
      from product.plans
      order by cticker;

   %if &fixedassets ne 0 %then %do;
      /* this is map from FA1,... to tickers for each plan(cticker) */

      proc sql;
      create table fa_map as
         select cticker,ticker as fatick,count(*) as lvl
         from product.fixedast a, product.plans  b
         where a.cticker=b.plan
         group by cticker, ticker
         order by cticker, ticker;

      data fa_map(drop=tickno maxfa);
      set fa_map end=eof;
      by cticker;
      retain tickno 0 maxfa;
      format faid $8.;
      if first.cticker then tickno=0;
      tickno = tickno + 1;
      if maxfa < tickno then maxfa=tickno;
      faid = 'FA'||put(tickno,1.);
      if eof then call symput("MAXFA",compress(put(maxfa,3.)));
      run;

      proc sql noprint;
      create table fat as
      select a.cticker, b.faid, a.ticker, a.wgt
      from product.fixedast a, fa_map b
      where a.cticker=b.cticker
      and a.ticker = b.fatick
      order by a.cticker,b.faid;


      data _pf_ds(drop=faid rename=(ticker=t1 wgt=w1));
      set fat(where=(faid='FA1'));
      run;

      %do i = 2 %to &maxfa;
         data fat2;
         set fat(where=(faid="FA&i"));
         run;

         proc sql noprint;
         create table _pf_ds as
            select a.*,b.ticker as t&i, b.wgt as w&i
            from _pf_ds a left join fat2 b on  a.cticker=b.cticker
            order by a.cticker;
```

```
      %end;


      data product.pfmaster;
      merge _pf_ds product.pfmaster;
      by cticker;
      run;

   %end;

   data product.pfmaster;
   set product.pfmaster;
   %do i = 1 %to &risklevs;
      risklev = &i;
      output;
   %end;
   run;

   proc sort data=product.pfmaster;
   by cticker risklev  %do i = 1 %to &maxfa; w&i %end;;
   run;


   data product.pfmaster;
   format pfid $8. pfname $20. pflname $40.;
   set product.pfmaster;
   pfid = "PF"||put(_n_,Z6.);
   pfname = trim(cticker)||'/'||put(risklev,1.);
   pflname = trim(pfname);
   %do i = 1 %to &maxfa;
      if w&i = . then w&i = 0;
      pfname = trim(pfname)||'/'||put(w&i*100,z2.);
      pflname = trim(pflname)||'/'||trim(t&i)||'/'||put(w&i*100,z2.);
   %end;
   run;

   proc sql noprint;
   create index pfname on product.pfmaster(pfname);

%mend;


%macro getmaxfa;

   %global maxfa;
   proc contents data=product.pfmaster(drop=pfid pfname pflname cticker proxyplan risk
lev) out=_m1 short noprint;
   run;
   proc sql noprint;
   select count(*)/2 into :maxfa
      from _m1;


%mend;


%macro pfid2mac(pfid=);
/* pfid to macros cticker risklev stktck1-stktckn stkwgt1-stkwgtn pfname pflname
*/

   /* get maximum number of fixed assets in a plan */
   %global maxfa cticker risklev pfname pflname;
   %local i;
   %getmaxfa;
   %do i = 0 %to &maxfa;
      %global stkwgt&i stktck&i;
   %end;

   %let stkwgt1 = 00;
   %let stktck1 = ;

   data _null_;
   set product.pfmaster(where=(pfid="&pfid")) end=eof;
   call symput("CTICKER", cticker);
   call symput("RISKLEV", compress(put(risklev,2.)));
   stklev0=0;
   %do i = 1 %to &maxfa;
      if t&i ne '' then do;
         stklev0 = stklev0 + 1;
         call symput("STKWGT&i",compress(put(w&i*100,z2.)));
         call symput("STKTCK&i",compress(t&i));
      end;
   %end;
   if eof then call symput("STKWGT0",compress(put(stklev0,2.)));
   run;
   %let stktck0 = &stkwgt0;
   %let pfname = &cticker/&risklev;
   %let pflname=&pfname;

   %do i = 1 %to &maxfa;
      %let pfname = &pfname/&&stkwgt&i;
      %let pflname = &pflname/&&stktck&i/&&stkwgt&i;
   %end;

%mend;

%macro pfma(pma=,nma=);

   %global &pma.0 &nma.0;

   data _master;
   set product.pfmaster(where=(proxyplan=cticker));
   run;


   proc contents data=_master(drop=pfid pfname pflname cticker risklev proxyplan )
   out=_m1 short noprint;
   run;
   proc sql noprint;
   select count(*)/2 into :maxfa
      from _m1;

   %ds2ma(in_ds=_master, var=pfname, ma=&nma, uniq=0);
   %ds2ma(in_ds=_master, var=pfid, ma=&pma, uniq=0);

%mend;


%macro pfnam2mac(pfname=);
/* pfname to macros cticker risklev stktck1-stktckn stkwgt1-stkwgtn pfid pflname
*/

   %global pfid pflname cticker risklev maxfa sumstklev ;
   %local i j p im2;

   data _null_;
   pfname = translate("&pfname",' ','/');
   call symput("P", pfname);
   run;
   %put pfname=&pfname, p=&p;
```

```
   %l2ma(list=&p, ma=_p);
   %let cticker = &_p1;
   %let risklev = &_p2;
   %let _p0 = %eval(&_p0 - 2);

   %let maxfa = &_p0;

   %do i = 0 %to &maxfa;
      %global stkwgt&i stktck&i;
   %end;
   %let stkwgt0=&maxfa;
   %let stktck0=&maxfa;

   proc sql noprint;
   select pfid,pflname
   %do i = 1 %to &_p0;
      ,t&i, put(100*w&i,z2.)
   %end;
   into :pfid, :pflname
   %do i = 1 %to &_p0;
      , :stktck&i, :stkwgt&i
   %end;
   from product.pfmaster
      where pfname = "&pfname";

   %let sumstklev = 0;

   %do i = 1 %to &stkwgt0;
      %let sumstklev = %eval(&sumstklev + &&stkwgt&i);
   %end;

%mend;



%macro m2filename(m_in=,m_out=);
   %global &m_out;
   data _null_;
   y = "&&&m_in";
   y = translate(y,'_','/');
   call symput("&m_out",y);
   run;
%mend;
```

```
%macro opto_pf;
/* optimize a portfolio
input:
%getenv must have been run first to set up the pf library and macros
product.plast(cticker,ticker) plan - assets
product.opop(lambda,kappa,linear,minval) optimizer run options
product.pfmaster(pfid,cticker,resklev,fal,...fan) - portfolio list with fixed asset le
vels
product.dtt(date,ticker1,...) return series for all assets&indexes used in this run
product.asset_tv(ticker,....) asset characteristics for all assets used in this run
pod!pop2_ini(RUNID,PFNAME,ticker,wgt) inital values
pod!pop2_aaconst(RUNID,PFNAME,var,target,range) asset allocation constraints
pod!pop2_tyconst(RUNID,PFNAME,typevar,typeval) asset type constraints
output:
pf.out_ctw(cticker,ticker,wgt) - model output portfolio
pod!pop2_pfq task status is updated
*/

   %put Optimizing Portfolio &pfid &pflname;

   %ds2macro(in_ds=product.opop); /* get optimizer options */

   %global covmat;

   %let covmat = ;

   proc sql;
   create table pf.plast as
      select "&pfid" as cticker, ticker, fname
      from product.plast
      where cticker = "&cticker"
      order by ticker;

   %global need2run status;

   %let need2run = 1;

   %getaacon(RUNID=&RUNID, PFNAME=&PFNAME, cticker=&cticker, risklev=&risklev, aaconst
=pf.aaconst); /* get asset allocation constraints */

   %mk_bendtt(in_tv=pf.plast,in_dtt=product.dtt,in_targ=pf.aaconst, out_dtt=pf.mdl_dtt
);

   %special_cases;


   %if &need2run eq 1 %then %do;

      %getopini(RUNID=&RUNID, PFNAME=&PFNAME, out_ctw=pf.initds);  /* get initial valu
es for optimizer */


      %gettycon(RUNID=&RUNID, PFNAME=&PFNAME, tyconst=pf.tyconst); /* get asset type c
onstraints */

      %getfacon(faconst=faconst); /* get fixed asset constraints */

      data pf.sconst;
      merge pf.tyconst faconst;
      by ticker;
      if wgt ne . then do;
         maxwgt=wgt;
         minwgt=wgt;
         end;
      run;



      %ret2tv(in_tv=pf.plast, in_dtt=pf.mdl_dtt);

      %taradj(inconst=pf.aaconst, outconst=pf.adjconst, in_dtt=product.dtt, fixed=faco
nst , in_tv= product.asset_tv);

      %if &status eq 0 %then %do;

         %special_constraints;

         /* first do no-minimum optimization */


         %opto_ctw(in_ctw=pf.initds, in_dtt=pf.mdl_dtt, bench=_BENCH, in_tv=product.as
set_tv,
                   lambda=&lambda, kappa=&kappa, linear=&linear, tconst=pf.adjconst, s
const=pf.sconst,
                   out_ctw=pf.ctw, minval=0,inquad=pf.inquad,outquad=pf.outquad,qoptio
ns=&qoptions,covmat=&covmat);

         %if %sysevalf(&status eq 0  and &minval > 0) %then %do;

            proc sql;
            select min(wgt)
               into :mwgt
               from pf.ctw
               where wgt >0;
            %if %sysevalf(&mwgt < &minval) %then %do;
               proc sql;
               create table initds as
                  select a.*,b.atype,c.wgt as wgto
                  from pf.initds a, product.asset_tv b, pf.ctw c
                  where a.ticker = b.ticker
                  and a.ticker = c.ticker
                  order by c.wgt desc, b.pvaluem;
               data initds(keep=cticker ticker wgt);
               set initds;
               retain ecount 0;
               retain fcount 0;
               if atype = 'EQUITY' then do;
                  ecount = ecount + 1;
                  if ecount <= 6 then output;
                  end;
                  else if atype = 'FOREIGN' then do;
                     fcount = fcount + 1;
                     if fcount <= 2 then output;
                     end;
                     else output;
               run;

               %ds2l(in_ds=initds, var=ticker, uniq=1, list=ticks);
               %put reduced universe = &ticks;

               %opto_ctw(in_ctw=initds, in_dtt=pf.mdl_dtt, bench=_BENCH, in_tv=product
.asset_tv,
                         lambda=&lambda, kappa=&kappa, linear=&linear, tconst=pf.adjco
nst, sconst=pf.sconst,
                         out_ctw=pf.ctw, minval=&minval,inquad=pf.inquad,outquad=pf.ou
tquad,qoptions=&qoptions,covmat=&covmat,maxiter=100);
               %end;
            %end;
```

```
        %if &status eq 0 %then %do;
           %if &whole ne 0 %then %do;
              %ctw2whole_pct(in_ctw=pf.ctw, in_dtt=pf.mdl_dtt, out_ctw=pf.mdl_ctw);
           %end;
           %else %do;
              data pf.mdl_ctw;
              set pf.ctw(where=(wgt >= .0000001));
              run;
           %end;
           %terr_ctw(in_ctw=pf.mdl_ctw,bench=_BENCH,in_dtt=pf.mdl_dtt,out_ds=pf.mdl_t
err,peryear=12,out_dtt=,rebal=1);
           %put Optimization succeeded;
           %setstatus(RUNID=&RUNID, PFNAME=&PFNAME, status=done);
        %end;
        %else %do;
           %put Optimization failed;
           %setstatus(RUNID=&RUNID, PFNAME=&PFNAME, status=fail);
        %end;
     %end;
     %else %do;
        %put New target not possible. Optimization abandoned;
        %setstatus(RUNID=&RUNID, PFNAME=&PFNAME, status=notappl);
     %end;
  %end;
%mend;

%macro setstatus(RUNID=, PFNAME=, status= );

  %dbexec(db=POD, sql=%bquote(pop2q_setstatus '&RUNID', '&PFNAME', '&status'));

%mend;


%macro getopini(RUNID=, PFNAME=, out_ctw=pf.initds);
/* get optimizer initial values
*/
   %dbgetpfx(db=pod,ds=initds)
      select ticker, wgt
      from pop2_ini
      where RUNID=%bquote('&RUNID')
      and PFNAME = %bquote('&PFNAME')
      order by ticker
      );
   %if %sysfunc(exist(initds)) %then %do;
      data &out_ctw;
      merge pf.plast initds;
      by ticker;
      if wgt eq . then wgt = 0;
      run;
   %end;
   %else %do;
      proc sql noprint;
      select count(*) into :astcount
         from pf.plast;

      data &out_ctw;
      set pf.plast;
      wgt=1/&astcount;
      run;
   %end;

%mend;

%macro getaacon(RUNID=, PFNAME=, cticker=, risklev=, aaconst=);
/* get asset allocation constraints
*/


   %dbgetpfx(db=POD,ds=&aaconst)
      select 1 as priority, var,target,range
      from pop2_aaconst
      where RUNID=%bquote('&RUNID')
      and PFNAME=%bquote('&PFNAME')
      union
      select 2 as priority, var,target,range
      from pop2_aaconst
      where RUNID=%bquote('&RUNID')
      and PFNAME=%bquote('&cticker/&risklev')
      union
      select 3 as priority, var,target,range
      from pop2_aaconst
      where RUNID=%bquote('&PRODUCT/PRODUCTION/DEFAULT')
      and PFNAME=%bquote('DEFAULT/&risklev')
      order by var, priority
      );

   data &aaconst(drop=irange);
   set &aaconst(rename=(range=irange));
   retain range;
   by var priority;
   if first.var then range = irange;
   minwgt = target - range;
   if minwgt < 0 then minwgt = 0;
   maxwgt = target + range;
   if maxwgt > 1 then maxwgt = 1;
   if last.var then output;
   run;

%mend;

data class2idx;
   format var ticker $8.;
input var $ ticker $;
cards;
BONDS _BOND
CASH _CASH_
FSTOCKS _EAFE
DSTOCKS _W5000
HYBONDS _ML_HYM
;


%macro aacon2ctw(aaconst=,out_ctw=);

   data &out_ctw(keep= var target rename=(target=wgt));
   set &aaconst(keep=var target where=(target ne 0)) end=eof;
   retain _stocks 0 _fstocks 0;
   if var eq 'STOCKS' then _stocks = target;
   else do;
      if var eq 'FSTOCKS' then _fstocks = target;
      output;
   end;
   if eof then do;
      var = 'DSTOCKS';
      target = _stocks - _fstocks;
      output;
   end;
   run;
```

```
   proc sql;
   create table &out_ctw as
      select "TARGET" as cticker length 8, b.ticker, a.wgt
      from &out_ctw a, class2idx b
      where a.var=b.var
      order by ticker;


%mend;



%macro gettycon(RUNID=, PFNAME=, tyconst=);
/* get asset type constraints
*/

   %dbgetpfx(db=pod,ds=_const)
      select 1 as priority, typevar,typeval,minwgt,maxwgt
      from pop2_tyconst
      where RUNID=%bquote('&RUNID')
      and PFNAME=%bquote('&PFNAME')
      union
      select 2 as priority, typevar,typeval,minwgt,maxwgt
      from pop2_tyconst
      where RUNID=%bquote('&RUNID')
      and PFNAME=%bquote('&cticker/&risklev')
      union
      select 3 as priority, typevar,typeval,minwgt,maxwgt
      from pop2_tyconst
      where RUNID=%bquote('&RUNID')
      and PFNAME=%bquote('')
      order by typevar,typeval,priority
      );
   data _const;
   set _const;
   by typevar typeval priority;
   if first.priority;
   run;

   %ds2ma(in_ds=_const, var=typevar, ma=_tvr, uniq=0, num=0);
   %ds2ma(in_ds=_const, var=typeval, ma=_tvl, uniq=0, num=0);
   %ds2ma(in_ds=_const, var=minwgt,  ma=_tmn, uniq=0, num=1);
   %ds2ma(in_ds=_const, var=maxwgt,  ma=_tmx, uniq=0, num=1);

   proc sql noprint;
   create table &tyconst(ticker char(8), minwgt num, maxwgt num);

   %local i;
   %do i = 1 %to &_tvr0;
      proc sql noprint;
      create table _const as
         select a.ticker,&&_tmn&i as minwgt, &&_tmx&i as maxwgt
         from pf.plast a, product.asset_tv b
         where a.ticker=b.ticker
         and b.&&_tvr&i = "&&_tvl&i";

      %dsappend(i=&i,in_ds=_const,out_ds=&tyconst);
   %end;
   proc sort data=&tyconst;
   by ticker;
   run;

   %ty_special_cases;

%mend;

%macro getfacon(faconst=);
/* get fixed asset constraints ticker,minwgt,maxwgt
*/

   %local i;

   data &faconst;
   format ticker $8. wgt best.;
   if &stktck0 > 0;
   %do i = 1 %to &stktck0;
      ticker = "&&stktck&i";
      wgt = &&stkwgt&i/100;
      minwgt=wgt;
      maxwgt=wgt;
      if ticker ne '' then output;
   %end;
   run;

%mend;


%macro mk_bendtt(in_tv=,in_dtt=,in_targ=,out_dtt=);
/* make benchmarks for the portfolios  */

   %ds2l(in_ds=&in_tv, var=ticker, list=plops );

   proc transpose data=&in_targ(keep=var target) out=fpp_targ(drop=_name_);
   id var;
   var target;
   run;

   %global TAR_FSTOCKS TAR_DSTOCKS TAR_STOCKS;

   data fpp_targ(rename=(stocks=stocks_ fstocks=fstocks_ bonds=bonds_ cash=cash_ hybon
ds=hybonds_));
   format stocks  fstocks bonds hybonds  cash 8.4;
   set fpp_targ end=eof;
   if eof then do;
      call symput("TAR_FSTOCKS",compress(put(fstocks,best.)));
      call symput("TAR_DSTOCKS",compress(put( stocks-fstocks,best.)));
      call symput("TAR_STOCKS",compress(put( stocks,best.)));
   end;
   run;

   data &out_dtt(keep=date _bench _EQBENCH &plops);
   merge fpp_targ &in_dtt;
   retain Tstocks_  Tfstocks_ Tbonds_ Thybonds_ Tcash_;
   array _ac(*) stocks_ fstocks_ bonds_ hybonds_ cash_;
   array _tac(*) Tstocks_  Tfstocks_ Tbonds_ Thybonds_ Tcash_;
   if _n_ eq 1 then do;
      do i = 1 to 5;
         if _ac[i] eq . then _ac[i] = 0;
         _tac[i] = _ac[i];
      end;
   end;
   _bench = (tstocks_-tfstocks_)*_w5000 + tfstocks_*_eafe + (tbonds_-thybonds_)*_bond
+thybonds_*_ml_hym + tcash_*_cash_ ;
   if tstocks_ > 0 then _EQBENCH=((tstocks_-tfstocks_)*_w5000 + tfstocks_*_eafe)/tstoc
ks_; else _eqbench = .;
   run;

%mend;
```

```
%macro objfn(in_ctw=,in_dtt=,bench=_BENCH,in_tv=product.asset_tv, objfn=,terrcon=,linc
on=,concon=,peryear=12);
/* objective function component values into macro variables
in_ctw(cticker,ticker,wgt) a single portfolio (cticker is ignored)
in_dtt(date,ticker1,ticker2,...) return series
bench - name of ticker that has benchmark return series
in_tv(ticker,&linear,...) asset characteristics (used to retrieve linear factor values
)
objfn   - name of macro to put objective function value into
terrcon - name of macro to put tracking error into
lincon  - name of macro to put &linear contribution into
concon  - name of macro to put concentration contribution into
peryear - number of periods per year for return series (used in annualizing tracking e
rror)
note that the variables &lambda, &kappa, &linear are assumed to be globally set
*/
   %global &objfn &terrcon &lincon &concon;
   %terr_ctw(in_ctw=&in_ctw,bench=&bench,in_dtt=&in_dtt,outmv=&terrcon,peryear=&peryea
r,rebal=1);

   proc sql noprint;
   select &lambda*&&&terrcon*&&&terrcon/&peryear+sum(a.wgt*b.&linear)+&kappa*sum(a.wgt
*a.wgt), sum(a.wgt*b.&linear), sum(a.wgt*a.wgt)
      into :&objfn, :&lincon, :&concon
      from &in_ctw a, &in_tv b
      where a.ticker=b.ticker;

%mend;
```

pop2_post.macro


```
%macro assemble_pf;
   /* assemble all model portfolios into a single ctw . Also assemble targets */

   %global pfid;

   %getenv;

   %pfma(pma=pf,nma=pn);

   %local i;
   proc sql;
   create table product.mdl_ctw    (cticker char(8), ticker char(8), wgt num);
   create table product.mdl_target (cticker char(8), ticker char(8), wgt num);

   %do i = 1 %to &pf0;
      %let pfid = &&pf&i;
      %let pfname = &&pn&i;
      %if %sysfunc(LIBNAME(pf,&prodbase/pf/&pfname)) eq 0 %then %do;
         %if %sysfunc(EXIST(pf.mdl_ctw)) %then %do;
            %put &pfid &pfname OK;
            proc append base=product.mdl_ctw data=pf.mdl_ctw force;
            run;
         %end;
         %else %put &pfid &pfname NO mdl_ctw;
         %if %sysfunc(EXIST(pf.adjconst)) %then %do;
            data x(keep= cticker var target rename=(var=ticker target=wgt));
            length cticker ticker $8;
            format wgt 6.2;
            retain cticker "&PFID";
            set pf.adjconst;
            run;
            proc append base=product.mdl_target data=x;
            run;
         %end;
/*          %else %put &pfid &pfname NO adjconst ******************************;*/
      %end;
      %else %put &pfid &pfname NO directory &prodbase/pf/&pfname **********************
*******;
   %end;

%mend;

%macro summ_pf(in_ctw=,out_dtr=,out_terr=);

   /* get benchmarks for risk levels */

   %ctwret(in_ctw=product.benches,in_dtt=product.dtt,out_dtt=bench_dtt);

   data product.dtt;
   merge product.dtt bench_dtt;
   by date;
   run;


   /* compute tracking error for each risk level separately */

   %local i ii;
   %let ii = 0;

   %do i = 1 %to 6;

      proc sql;
      create table _levctw as
         select a.cticker,a.ticker,a.wgt
          from &in_ctw a, product.pfmaster b
          where a.cticker=b.pfid
          and b.risklev = &i;
      %if %eval(&sqlobs > 0) %then %do;
         %let ii = %eval(&ii+1);

         %terr_ctw(in_ctw=_levctw,bench=_BENCH&i,in_dtt=product.dtt,out_ds=terr,peryea
r=12,out_dtt=ctw_dtt,rebal=1);
         %dtt2dtr(in_dtt=ctw_dtt,out_dtr=ctw_dtr);
         %dsappend(i=&i,in_ds=terr,out_ds=&out_terr);
         %dsappend(i=&i,in_ds=ctw_dtr,out_ds=&out_dtr);
      %end;

   %end;
   proc sort data=&out_dtr;
   by ticker date;
   run;
   proc sort data=&out_terr;
   by ticker;
   run;


%mend;

%macro aaconst2benches;

   %dbgetpfx(db=pod,ds=aaconst)
   select PFNAME,var,target,range
      from pop2_aaconst
      where RUNID = %bquote('&PRODUCT/PRODUCTION/DEFAULT')
      order by PFNAME
      );


   %l2wtran(in_ds=aaconst,var=target,by=PFNAME,id=var);

   data aaconst(keep=cticker dstocks  igbonds cash fstocks hybonds
                rename=(dstocks=_W5000 fstocks=_EAFE igbonds=_BOND hybonds=_ML_HYM cas
h=_CASH));
   format cticker $8. dstocks fstocks stocks bonds igbonds hybonds cash other   z4.02;
   set aaconst;
   dstocks=stocks-fstocks;
   igbonds=bonds-hybonds;
   cticker = "_BENCH"||put(substr(PFNAME,9,1),1.);
   run;

   %w2ltran(in_ds=aaconst,var=wgt,by=cticker,id=ticker,out_ds=product.benches);

   data product.benches;
   format cticker $8. ticker $8. wgt z4.2;
   set product.benches(where=(wgt ne 0));
   run;


%mend;

%macro pop2_postprocess(prune=0);

   %aaconst2benches;

   %assemble_pf;
   %if &prune ne 0 %then %prune_ctw;

   %summ_pf(in_ctw=product.mdl_ctw,out_dtr=product.mdl_dtr,out_terr=product.mdl_terr);
```

```
%mend;


%macro prune_ctw;

   proc sort data=product.mdl_ctw;
   by cticker ;
   run;

   proc sql;
   create table made_pfs as
      select distinct a.pfid,a.cticker,risklev,w1 as stklev
      from product.pfmaster a, product.mdl_ctw b
      where b.cticker=a.pfid
      order by a.cticker,risklev,stklev;


   data keep_pfs(keep=pfid) drop_pfs;
   retain last_rs next_s lastkept 0;
   set made_pfs;
   by cticker risklev stklev;
   if first.risklev then next_s = 0;
   if (risklev eq 1 or last_rs ge stklev) and abs(stklev-next_s)<.00001 then do;
     output keep_pfs;
     lastkept = stklev;
     next_s = stklev+.1;
   end;
   else output drop_pfs;
   if last.risklev then last_rs = lastkept;
   run;

   proc sort data=keep_pfs;
   by pfid;
   run;


   data product.mdl_ctw product.raw_ctw;
   merge product.mdl_ctw keep_pfs(rename=(pfid=cticker) in=ink);
   by cticker;
   if ink then output product.mdl_ctw ;
   output product.raw_ctw;
   run;

   proc sql;
   create table product.mdl_target as
      select a.*
      from product.mdl_target a, keep_pfs b
      where a.cticker=b.pfid;


%mend;
```

pop2_pre.macro


```
%macro default2aaconst;

   %dbexec(db=pod, sql=delete pop2_aaconst where RUNID=%bquote('&RUNID'));

   %dblibname(db=POD);

   proc sql noprint;
   select count(*)
      into :rancount
      from pod.pop2_aaconst
      where RUNID = "&runid";

   %if %eval(&rancount eq 0) %then %do;

      proc sql noprint;
      create table aaconst as
         select "&runid" as runid, b.PFNAME,a.var,a.target,a.range
         from pod.pop2_aaconst a, product.pfmaster b
         where a.runid="&product/PRODUCTION/DEFAULT"
         and substr(a.PFNAME,9,1) = put(b.risklev,z1.)
         order by b.PFNAME,a.var;

      proc append base=pod.pop2_aaconst(bulkcopy=yes) data=aaconst;
      run;
   %end;



%mend;
```

```
%macro mk_asset_tv(in_t=,use_mstar=1,use_fame=1,use_fprs=0,otherds=);
/* make product.asset_tv from reference datasets
in_t(ticker) live tickers in product
use_mstar, use_fame,use_fprs - flags indicating whether corresponding
data source should be used in building asset_tv
- Loads reference data used in optimization.
- Derives asset class percentages
- Appends _CASH_ and _GIC_
*/

   %if &otherds ne %then %do;
      %tvfilter(in_tv=&otherds,in_t=&in_t, out_tv=other_tv);
   %end;

   %if &use_mstar ne 0 %then %do;
      %tvfilter(in_tv=baseds.premast,  in_t=&in_t,  out_tv=mstar_tv);
   %end;

   %if &use_fame ne 0 %then %do;
      %tvfilter(in_tv=baseds.eq_mstr,  in_t=&in_t,  out_tv=fame_tv);
   /*       %v2vv(in_ds=fame_tv,var=industry);  */
   %end;

   %if &use_fprs ne 0 %then %do;
      %tvfilter(in_tv=product.fprs_tv, in_t=&in_t,  out_tv=fprs_tv);
   %end;

   data product.asset_tv(keep=ticker fname stocks dstocks fstocks bonds igbonds hybond
s cash other mcatn fobjn src atype
                         DURABL ENERGY FINANC HEALTH INDUST RETAIL SERVIC STAPLE TECHN
O UTILIT);
   format ticker $8.  fname $80. stocks dstocks fstocks bonds cash other 6.4 src $5. a
type $8.;
   set
      %if &use_fprs ne 0 %then fprs_tv(in=infprs rename=(f_name=fname));
      %if &use_mstar ne 0 %then mstar_tv(in=inmstar);
      %if &use_fame  ne 0 %then fame_tv(in=infame rename=(issuer=fname forgn=fstocks)
);
     %if &otherds ne %then other_tv(in=inother);
      end=eof;
   %if &use_mstar eq 0 %then retain inmstar 0;;
   %if &otherds ne %then retain inother 0;;

   array _x  DURABL ENERGY FINANC HEALTH INDUST RETAIL SERVIC STAPLE TECHNO UTILIT;
   %if &use_mstar ne 0 or &otherds ne  %then %do;
      if inmstar or inother then do;
         do over _x;
            _x = _x/100;
            end;
         src="MSTAR";
         if (GovAgCy+AAA+AA+A+BBB+BB+B+BELOWB+NRNA) ne 0 then
            hybonds = bonds * ( (BB+B+BELOWB+NRNA)/(GovAgCy+AAA+AA+A+BBB+BB+B+BELOWB+N
RNA) );
         if hybonds = . then hybonds = 0;
         totwgt=sum(bdomest,bforeig,edomest,eforeig,other,cash,prefrd,convert);
         stocks=(edomest+eforeig)/totwgt;
         dstocks=edomest/totwgt;
         fstocks=eforeig/totwgt;
         bonds=(sum(bdomest,bforeig)/totwgt);
         cash=(cash/totwgt);
         other=1-sum(stocks,bonds,cash);
         end;
   %end;
   %if &use_fame ne 0 %then %do;
      if infame then do;
         do over _x;
            if _x eq . then _x = 0;
         end;
         src="FAME";
         bonds=bonds/100;
         hybonds=0;
         cash=cash/100;
         other=other/100;
         fstocks = fstocks/100;
         stocks = stocks/100;
         dstocks = stocks - fstocks;
         if ( (bonds+cash) gt .90 ) then atype = "BOND";
         else if fstocks gt .70 then atype = 'FOREIGN';
         else atype = 'EQUITY';
         end;
   %end;
   %if &use_fprs ne 0 %then %do;
      if infprs then do;
         do over _x;
            _x = _x/100;
            end;
         src="FPRS";
         stocks=stocks/100;
         fstocks=(stocks * foreign/100);
         dstocks=stocks - fstocks;
         bonds=bonds/100;
         cash=cash/100;
         other=other/100;
         if (GovAgCy+AAA+AA+A+BBB+BB+B+BELOWB+NRNA) ne 0 then
            hybonds = bonds * ( (BB+B+BELOWB+NRNA)/(GovAgCy+AAA+AA+A+BBB+BB+B+BELOWB+N
RNA) );
         if hybonds = . then hybonds = 0;
         if ( (bonds+cash) gt .90 ) then atype = "BOND";
         else if fstocks gt .70 then atype = 'FOREIGN';
         else atype = 'EQUITY';
         end;
   %end;
   igbonds = bonds - hybonds;
   output;
   if eof then do;
      fname = 'Money Market Fund';
      do over _x;
         _x = .;
      end;
      ticker='_CASH_';
      cash=1;
      stocks=0;
      dstocks=0;
      fstocks=0;
      bonds=0;
      hybonds=0;
      igbonds=0;
      other=0;
      atype="CASH";
      mcatn  ='Money Market Fund';
      fobjn  ='Money Market Fund';
      src = '';
      output;
      fname = 'Stable Value Fund';
      ticker='_GIC_';
      do over _x;
         _x = .;
      end;
      cash=.2;
      bonds=.8;
```

```
      stocks=0;
      dstocks=0;
      fstocks=0;
      hybonds=0;
      igbonds=.8;
      other=0;
      atype="GIC";
      mcatn  ='Stable Value Fund';
      fobjn  ='Stable Value Fund';
      src = '';
      output;
      end;
   run;

   proc sort data=product.asset_tv;
   by ticker src;
   run;

   %mk_atype(product.asset_tv);

%mend;


/* calculates asset type from Morningstar Categories */

data atype;
format atype $8. mcatn $32.;
input atype $1-8 mcatn $10-41;
cards;
BOND     Intermediate-Term Bond
BOND     Long-Term Bond
BOND     Short-Term Bond
EQUITY   Convertibles
EQUITY   Domestic Hybrid
EQUITY   Diversified Pacific/Asia Stock
EQUITY   Diversified Emerging Markets
EQUITY   Europe Stock
FOREIGN  Foreign Stock
BOND     Intermediate Government
BOND     Long Government
BOND     Short Government
BOND     High Yield Bond
EQUITY   International Bond
EQUITY   International Hybrid
EQUITY   Japan Stock
EQUITY   Large Blend
EQUITY   Large Growth
EQUITY   Latin America Stock
EQUITY   Large Value
EQUITY   Mid-Cap Blend
EQUITY   Mid-Cap Growth
MUNI     Muni National Interm
MUNI     Muni National Long
MUNI     Muni Short
EQUITY   Multisector Bond
EQUITY   Mid-Cap Value
EQUITY   Pacific/Asia ex-Japan Stock
EQUITY   Small Blend
EQUITY   Specialty--Communications
EQUITY   Specialty--Financial
EQUITY   Small Growth
EQUITY   Specialty--Health
MUNI     Muni Single State Intermediate
MUNI     Muni Single State Long
EQUITY   Specialty--Natural Resources
EQUITY   Specialty--Precious Metals
EQUITY   Specialty--Real Estate
EQUITY   Specialty--Technology
EQUITY   Specialty--Utilities
EQUITY   Small Value
EQUITY   Ultrashort Bond
FOREIGN  World Stock
CASH     Money Market
;

%macro mk_atype(asset_tv);

   proc sort data=&asset_tv;
   by mcatn;
   proc sort data=atype;
   by mcatn;
   run;

   data &asset_tv(drop=newatype);
   merge &asset_tv(in=inast ) atype(in=inat rename=(atype=newatype));
   by mcatn;
   if inast;
   if inat then atype=newatype;
   run;

   proc sort data=&asset_tv;
   by ticker;
   run;

%mend;

%macro objfilter(in_ds=,out_ds=);

   data &out_ds;
   set  &in_ds;
   if (mcatn ne '') then do;
      if ( fobjn eq 'Balanced' ) and ( mcatn eq 'Domestic Hybrid' ) then output;

      else if ( fobjn in ('Aggressive Growth','Equity-Income','Growth',
                          'Growth and Income','Small Company') ) then do;
           if ( mcatn in ('Large Blend','Large Growth','Large Value',
                          'Mid-Cap Blend','Mid-Cap Growth','Mid-Cap Value',
                          'Small Blend','Small Growth','Small Value') )
              then do;
                 %* cash limit!!! ;
                 if ( cash lt 13 ) then output;
              end;
      end;
      else if ( fobjn in ('World Stock','Foreign Stock') )
         then do;
            if ( mcatn in ('World Stock','Foreign Stock') ) then output;
         end;
      else if ( index( fobjn , 'Govt Bond' ) eq 1 ) or
              ( index( fobjn, 'Corp Bond' ) eq 1 )
      then do;
         if ( index( mcatn, 'High Yield' ) eq 1 ) or
            ( index( mcatn, 'Intermediate' ) eq 1 ) or
            ( index( mcatn, 'Long' ) eq 1 ) or
            ( index( mcatn, 'Short' ) eq 1 ) or
            ( index( mcatn, 'Ultrashort' ) eq 1 )
         then output;
      end;
      %* set cash aside ;
         else if ( ticker in ("_CASH_","_GIC_")) then output;
      %* set all supplement options aside ;
```

```
   end;
   else output;
   run;


%mend;

%macro mk_dtt(from_dt=, to_dt=,asset_tv=product.asset_tv,stock=1,fund=1,other_dtt=);
/* make product.dtt, product.dtr, product.missret product.idx_dtt from reference datas
ets
in_t(ticker) live tickers in product
use_mstar, use_fame,use_fprs - flags indicating whether corresponding
data source was used in building asset_tv
*/


   %let indexes = _BOND _CASH_ _W5000 _R1000  _R2000  _R1000V  _R2000V  _R1000G  _R200
0G  _R3000 _R3000G _R3000V _ML_HYM  _LB_TR _SB_10TR _SB_TR _SB_1YTR _LB_MBS _MS_FARE _
MS_EURO _MS_AUST _MS_NA _IF_LAM _USDTW _LB_MUNI _LB_10MU _LB_14MU _EAFE _LB_13GV;
   %m_idxret(from_dt=&from_dt, to_dt=&to_dt, tickers=&indexes,out_dtr=, out_dtt=produc
t.idx_dtt);

   %if &stock ne 0 %then %do;
      %m_stkret(from_dt=&from_dt, to_dt=&to_dt, in_tv=&asset_tv, out_dtt=stk_dtt);
      %let stk_dtt = stk_dtt;
   %end;
   %else %let stk_dtt = ;

   %if &fund ne 0 %then %do;
      %m_fndret(from_dt=&from_dt, to_dt=&to_dt, in_tv=&asset_tv, out_dtt=fnd_dtt);
      %let fnd_dtt = fnd_dtt;
   %end;
   %else %let fnd_dtt = ;

   data product.dtt;
   merge product.idx_dtt &stk_dtt &fnd_dtt  &other_dtt(where=(date <= &refdate and dat
e >= &retfrom));
   by date;
   _GIC_ = _LB_13GV;
   run;

   %dtt2dtr(in_dtt=product.dtt,out_dtr=product.dtr);

   %dtr_hole(in_tv=product.plast, in_dtr=product.dtr, out_ds= product.missret);


%mend;

%macro aclass2idx(in_ds=,out_ds=,var=,by=,id=);

   %if &out_ds eq %then %let out_ds=&in_ds;

   %if &var ne %then %do;

      %l2wtran(in_ds=&in_ds,out_ds=&out_ds,var=&var,by=&by,id=&id);
      data &out_ds(keep = dstocks fstocks igbonds hybonds cash &by rename=(dstocks=_W5
000 fstocks=_EAFE igbonds=_BOND hybonds=_ML_HYM cash=_CASH));
      set &out_ds;
      if hybonds eq . then hybonds = 0;
      if stocks ne . then dstocks=stocks-fstocks;
      if bonds ne . then igbonds=bonds-hybonds;
      run;
      %w2ltran(in_ds=&out_ds,var=&var,by=&by,id=&id);
   %end;
   %else %do;
      data &out_ds(keep = dstocks fstocks igbonds hybonds cash rename=(dstocks=_W5000
fstocks=_EAFE igbonds=_BOND hybonds=_ML_HYM cash=_CASH));
      set &in_ds;
      if hybonds eq . then hybonds = 0;
      if stocks ne . then dstocks=stocks-fstocks;
      if bonds ne . then igbonds=bonds-hybonds;
      run;
   %end;


%mend;
```

```
/* quadratic optimization
macros that might simplify use of proc nlp
*/


%macro quad(in_tv=, covmat=, lambda=1, kappa=0, linear=, initval=, gconst=, tconst=, s
const=, qoptions=, inquad=inquad, outquad=,title=);
 /* quadratic optimization
  in_tv(ticker,var1,...) - asset list of assets to be optimized. required.
  covmat - covariance matrix of assets in in_tv. required.
  kappa - concentration multiplier. defaults to 0.
  lambda - covariance multiplier. defaults to 1.
  linear - name of var to get linear term of objective function from. optional
  initval(ticker,wgt) - asset-weight ds of initial values. optional
  tconst(var,minwgt,maxwgt) - total constraints dataset. optional
  sconst(ticker,minwgt,maxwgt) - specific constraints dataset. optional
  qoptions - options for proc nlp command
  inquad - name of resultant dataset used for proc nlp input.
  outquad - proc nlp output dataset. optional: if absent proc nlp is not called.
  title - optional title for lst file entry
  */
   %local i j;
   %ds2ma(in_ds=&in_tv,var=ticker, ma=al);

   data &inquad(keep = _type_ _name_ %do i = 1 %to &al0; &&al&i %end; _rhs_);
   format _type_ $8. _name_ $8. _vname $8.;
   set &covmat(where=(_type_ = 'COV'
                      and _name_ in ("&al1" %do i = 2 %to &al0; ,"&&al&i" %end;))
               ) end = eof;
   retain _rhs_ . _vname '';
   _type_ = 'QUAD';
   %do i = 1 %to &al0;
      &&al&i = &&al&i * 2 * &lambda;
      %if &kappa ne 0 %then %do;
         call vname(&&al&i,_vname);
         if _name_ = _vname then &&al&i = &&al&i + 2 * &kappa;
      %end;
   %end;

   output;
   %if &gconst eq %then %do;
      if eof then do;
         %do i =1 %to &&al0;
            &&al&i = 1;
         %end;
         _type_ = 'eq';
         _name_ = 'WGT';
         _rhs_ = 1;
         output;
      end;
   %end;
   run;

   %if &gconst ne %then %do;
      proc append base=&inquad data=&gconst force;
      run;
   %end;

   %if &linear ne %then %do;
      proc sql noprint;
      create table _q1 as
         select 'LINEAR   ' as _type_, . as _RHS_,    ticker, &linear as _linear
         from &in_tv ;

      proc transpose data=_q1 out=_q1;
      var _linear;
      id ticker;
      by _type_ _RHS_ ;

      data &inquad;
      set &inquad _q1;
      run;

   %end;

   %if &sconst ne %then %do;
      %s2qconst(ma=al,sconst=&sconst,qconst=tsconst);
      proc append base=&inquad data=tsconst force;
      run;
   %end;

   %if &initval eq %then %do;

      %let initval = _q1;
      %if &sconst eq %then %do;

         data _q2;
         set &sconst(in=in1) &in_tv(in=in2) ;
         if in1 then ord = 1;
         else do;
            ord=2;
            minwgt=0;
            maxwgt=1;
            end;

         proc sort data=_q2;
         by ticker ord;

         data _q2;
         set _q2;
         by ticker ord;
         if first.ticker;
         run;

         proc sql noprint;
         select sum(maxwgt-minwgt), 1-sum(minwgt)
            into :_play, :_add
            from _q2;

/*         %put initval comps: &_play/&_add; */
         data _q1(keep=ticker wgt);
         set _q2;
         wgt = &_add*(maxwgt-minwgt)/&_play + minwgt;
         run;
      %end;
      %else %do;
         data _q1;
         format ticker $8.;
         %do i = 1 %to &al0;
            ticker = "&&al&i";
            wgt = 1 / &al0;
            output;
         %end;
      %end;
   %end;

   proc transpose data=&initval out=_q1;
   var wgt;
   id ticker;
   run;
```

```
   data _q1;
   format _name_ $8.;
   set _q1;
   _type_ = 'PARMS   ';
   _RHS_ = .;
   proc append base=&inquad data=_q1 force;
   run;



   %if &tconst ne %then %do;
      %t2qconst(in_tv=&in_tv, tconst=&tconst, qconst=ttconst);
      proc append base=&inquad data=ttconst force;
      run;
   %end;

   %if &outquad ne %then %do;
      %if "&title" ne "" %then %do;
         data _null_;
         title "&title";
         run;
      %end;
      %callnlp(ma=al,inquad=&inquad,qoptions=&qoptions,outquad=&outquad);
   %end;

   %trash(_q1 _q2);

%mend;

%macro callnlp(ma=,inquad=inquad,qoptions=,outquad=outquad);
   %local i;

   proc nlp inquad=&inquad outest=&outquad cov=2 &qoptions;
   min;
   parms %do i = 1 %to &&&ma.0; &&&ma&i %end; ;
   run;

   %global objfnval;
   proc sql noprint;
   select _rhs_
      into :objfnval
      from &outquad
      where _type_ = 'PARMS';
   %let objfnval = &objfnval;
   %if "&objfnval" eq "." %then %do;
      %global ziter;
      %if &ziter eq %then %let ziter = 1;
      %else %let ziter = %eval(&ziter + 1);
      data _inq&ziter;
      set &inquad;
      run;
      %put failed on iter &ziter;
   %end;

%mend;


%macro q2ctw(outquad=outquad, out_ctw=, cticker=PARMS);
 /* converts nlp output to asset weight list
  input
  outquad - output of proc nlp, contains one line with _type_ = "PARMS"
  output
  ctw(cticker,ticker,wgt) - these are the values of parameters
  */
   %global ostatus objfnval;
   %let objfnval = .;
   %let ostatus = 1;

   proc sql;
   select _rhs_ into :objfnval
      from &outquad where _type_ = 'PARMS';

   %if "&objfnval" eq "" %then %let objfnval = .;

   %if "&objfnval" ne "." %then %do;

      data &out_ctw(drop=_type_ _rhs_);
      format cticker $8.;
      set &outquad(drop= _name_ _iter_ _tech_ where=(_type_='PARMS'));
      cticker = "&cticker";
      call symput("OBJFNVAL",put(_rhs_,best.));
      if _rhs_ ne . then do;
         output;
         call symput("OSTATUS","0");
      end;
      else do;
         call symput("OSTATUS","1");
      end;
      run;
   %end;

   %if &OSTATUS eq  0 %then %do;
      proc transpose data=&out_ctw out=&out_ctw(rename=(_name_=ticker col1=wgt));
      by cticker;
      run;
   %end;
   %else %do;
      proc sql noprint;
      create table &out_ctw(cticker char(8), ticker char(8), wgt num);
      %let objfnval = .;
       %put NO SOLUTION;
   %end;

%mend;


%macro s2qconst(ma=, sconst=, qconst=);
 /* transforms sconst to format suitable for nlp
  also resolves overlapping constraints, choosing
  the tightest
  (to be used only by %quad)
  */

   %local i;
   proc sort data=&sconst(keep = ticker maxwgt) out=_q3;
   by ticker maxwgt;
   data _q3;
   set _q3;
   by ticker maxwgt;
   if first.ticker;

   proc transpose data=_q3 out=_q4;
   var maxwgt;
   id ticker;

   data _q4;
   format  _type_ $8. _name_ $8.;
   set _q4;
   _type_ = 'upperbd ';
```

```
   _name_ = 'UPPER';
   _RHS_ = .;
   %do i = 1 %to &&&ma.0;
      if &&&ma&i eq . then &&&ma&i = 1;
   %end;
   run;

   proc sort data=&sconst(keep = ticker minwgt) out=_q5;
   by ticker descending minwgt;
   data _q5;
   set _q5;
   by ticker descending minwgt;
   if first.ticker;

   proc transpose data=_q5 out=_q6;
   var minwgt;
   id ticker;
   data _q6;
   format  _type_ $8. _name_ $8.;
   set _q6;
   _type_ = 'lowerbd ';
   _name_ = 'LOWER';
   _RHS_ = .;
   %do i = 1 %to &&&ma.0;
      if &&&ma&i eq . then &&&ma&i = 0;
   %end;
   run;

   data &qconst(keep = _type_ _name_ _rhs_ %do i = 1 %to &&&ma.0; &&&ma&i %end;);
   set _q4 _q6;
   run;

   %trash(_q3 _q4 _q5 _q6);

%mend;



%macro t2qconst(in_tv=, tconst=, qconst=);
 /* transforms tconst into format suitable for nlp
  (to be used only by %quad)
  */

   %local i;
   proc sql noprint;
   select count(*)
      into :cnt
      from &tconst;
   %let cnt=&cnt;

   proc sql noprint;
   select var, minwgt,maxwgt
      into  :vname1 - :vname&cnt,
         :minwgt1 - :minwgt&cnt,
         :maxwgt1 - :maxwgt&cnt
      from &tconst;

   data _q7;
   format _name_ $8. ticker $8. wgt 15.8 _type_ $8. _rhs_ 15.8;
   if 1=2;
   _name_='';
   ticker = '';
   wgt = .;
   _type_ = '';
   _rhs_ = .;
   run;

   %do i = 1 %to &cnt;
      %let &&vname&i=&&vname&i;
/*       %put namminmax - &&vname&i,&&minwgt&i,&&maxwgt&i;*/

      %if "&&minwgt&i" ne ""  and "&&minwgt&i" ne "." %then %do;
         data _q7;
         set _q7 &in_tv(in=inb keep=ticker &&vname&i rename=(&&vname&i=wgt ));
         if inb then do;
            _name_ = "&&vname&i";
            _type_ = 'ge';
            _rhs_ = &&minwgt&i;
          end;
         run;
      %end;

      %if  "&&maxwgt&i" ne "" and "&&maxwgt&i" ne "." %then %do;
         data _q7;
         set _q7 &in_tv(in=inb keep=ticker &&vname&i  rename=(&&vname&i=wgt));
         if inb then do;
            _name_ = "&&vname&i";
            _type_ = 'le';
            _rhs_ = &&maxwgt&i;
          end;
         run;
      %end;
   %end;

   proc sort data=_q7;
   by _name_ _type_ _rhs_;
   run;

   proc transpose data=_q7 out=&qconst(drop=_label_);
   by _name_ _type_ _rhs_;
   id ticker;
   var wgt;
   run;

   %trash(_q7);

%mend;


%macro mintrack(ticker=,in_dtt=,in_tv=, factors=,bench=,out_ds=,peryear=252, out_ctw=)
;
/* minimize tracking error of an asset vs a benchmark
ticker - ticker of resultant portfolio
in_dtt(date,&ticker,&bench,...) return series
choose in_tv | factors
in_tv(ticker,...) assets in portfolio
factors - list of assets in portfolio
bench - ticker of benchmark asset
out_ds(cticker,factor1,..,factorn,trackerr) - optimized portfolio
peryear - dates per year (252 - for daily returns, 12 for monthly, 1 for annual)
out_cwt(cticker,ticker,wgt) - ctw form of optimized portfolio
*/
   %local i;

   %if &out_ctw eq  %then %let out_ctw=_ctw;
   %if &in_tv ne %then %do;
      %ds2ma(in_ds=&in_tv,var=ticker,ma=fa);
   %end;
   %else %do;
      %l2ma(list=&factors,ma=fa);
```

```
      %l2ds(list=&factors,var=ticker,out_ds=_mt);
      %let in_tv = _mt;
   %end;

   data dtt(drop = &bench);
   set &in_dtt(keep = date &bench %do i = 1 %to &fa0; &&fa&i %end;);
   array _tick %do i = 1 %to &fa0; &&fa&i %end;;
   do over _tick;
      _tick = _tick - &bench;
   end;
   run;

   %put building covariance matrix for &fa0 assets;

   proc corr cov data=dtt outp=_covar nosimple noprint;
   var %do i = 1 %to &fa0; &&fa&i %end;;
   run;

   %put calling nlp;

   data quadin(keep=_type_ _name_ %do i = 1 %to &fa0; &&fa&i %end; _rhs_);
   length _type_ _name_ $8;
   set _covar(keep=_type_ _name_ %do i = 1 %to &fa0; &&fa&i %end; where=(_type_ eq 'CO
V')) end=eof;
   array _tick %do i = 1 %to &fa0; &&fa&i %end;;
   _type_ = 'QUAD';
   _rhs_ = .;
       do over _tick;
         _tick = _tick * 20000;
       end;
   output;
   if eof then do;
      do over _tick;
         _tick = 1;
         end;
      _type_ = 'eq';
      _name_ = 'WGT';
      _rhs_ = 1;
      output;
      _type_ = 'lowerbd ';
      _name_ = 'LOWER';
      _RHS_ = .;
      do over _tick;
         _tick = 0;
         end;
      output;
      _type_ = 'upperbd ';
      _name_ = 'UPPER';
      _RHS_ = .;
      do over _tick;
         _tick = 1;
         end;
      output;
      _type_ = "PARMS";
      _name_ = "PARMS";
      do over _tick;
         _tick = 1/&fa0;
         end;
      output;
   end;
   run;

   proc nlp inquad=quadin outest=outquad cov=2;
   min;
   parms %do i = 1 %to &fa0; &&fa&i %end;;
   run;
   proc sql noprint;
   select _rhs_
      into :objfnval
      from outquad
      where _type_ = 'PARMS';
   %let objfnval = &objfnval;

   %put cleaning up &objfnval;

   data &out_ctw(drop=_type_);
   set outquad(drop= _name_ _iter_ _tech_ _rhs_ where=(_type_='PARMS' ));
   cticker="&ticker";
   run;

   %w2ltran(in_ds=&out_ctw,out_ds=&out_ctw,var=wgt,id=ticker,by=cticker);
   data &out_ctw;
   set &out_ctw(where=(wgt > 1E-15));
   run;



   %if &out_ds ne %then %do;
      %terr_ctw(in_ctw=&out_ctw,in_dtt=&in_dtt,bench=&bench,peryear=&peryear,out_ds=_t
err,rebal=1);

      data &out_ds(drop=_type_ _rhs_);
      merge outquad(drop= _name_ _iter_ _tech_ where=(_type_='PARMS')) _terr;
      if _rhs_ ne .;
      run;
   %end;

%mend;

%macro terr_ctw(in_ctw=,bench=,in_dtt=,out_ds=,outmv=,peryear=252,out_dtt=,rebal=1);
/* compute tracking errors of portfolios
in_ctw(cticker,ticker,wgt) - portfolios
bench - ticker of benchmark, blank computes std of portfolios
in_dtt(date,ticker,ticker,...) return series containing benchmark and portfolio holdin
gs returns
out_ds(ticker,trackerr) - tracking errors of portfolios
out_dtt(date,cticker,cticker,...) return series of ctickers
peryear - dates per year (252 for daily, 12 for monthly returns)
  rebal - nonzero to indicate that the portfolios are rebalanced to the initial weight
s
          each date.
outmv - macro variable to put tracking error into.  For this option, there must be onl
y one portfolio in &in_ctw
*/
   %if &out_dtt eq %then %let out_dtt=_dtt;
   %if &bench eq %then %let bench = 0;

   %ctwret(in_ctw=&in_ctw,out_dtt=&out_dtt,in_dtt=&in_dtt,zeronull=0,rebal=&rebal);
   %dtt2dtr(in_dtt=&out_dtt,out_dtr=_dtr);
   %if &out_ds ne %then %do;
      proc sql noprint;
      create table &out_ds as
         select ticker, std(return-&bench)*sqrt(&peryear) as trackerr
         from _dtr a, &in_dtt b
         where a.date=b.date
         group by ticker
         order by ticker;
   %end;
   %if &outmv ne %then %do;
      %global &outmv;
```

```
      proc sql noprint;
      select std(return-&bench)*sqrt(&peryear)
         into :&outmv
         from _dtr a, &in_dtt b
         where a.date=b.date;
      %put tracking error = &&&outmv;
   %end;

%mend;

%macro terr(ticker=,in_dtt=,in_dtr=,bench=_R3000,outmv=trackerr,peryear=252);
/* compute tracking error of an asset
ticker - asset to compute tracking error for
bench - ticker of benchmark
choose(in_dtt | in_dtr)
in_dtt(date,&ticker,&bench) return series containing benchmark and asset returns
in_dtr(date,ticker,return) return series containing benchmark and asset returns
outmv - name of macro variable to put tracking error into
peryear - dates per year (252 for daily, 12 for monthly returns)
*/
   %local _terr;

   %if &in_dtt ne %then %do;
      proc sql noprint;
      select std(&ticker-&bench)*sqrt(&peryear)
         into :_terr
         from &in_dtt b;
   %end;
   %else %do;
      proc sql noprint;
      select std(a.return-b.return)*sqrt(&peryear)
         into :_terr
         from &in_dtr a, &in_dtr b
         where a.ticker = "&ticker"
         and b.ticker = "&bench"
         and a.date=b.date;
   %end;

   %if &outmv ne %then %do;
      %global &outmv;
      %let &outmv = &_terr;
   %end;
   %else %put Tracking Error = &_terr;

%mend;

%macro dtt2covm(in_dtt=,out_covm=);
/* return series to covariance matrix
*/

   %dsv2ma(in_ds=&in_dtt,drop=date,ma=_qm);
   %let _qv = %ma2l(ma=_qm);
   proc corr cov data=&in_dtt outp=&out_covm nosimple noprint;
   var &_qv;
   run;

%mend;

%macro mksconst(i=1, tickers = , minwgt =, maxwgt =, out_ds=);
/* make sconst
*/
   %local j o;
   %l2ma(list=&tickers, ma=_c);

   %if &i eq 1 %then %do;
      %let o = &out_ds;
   %end;
   %else %do;
      %let o = _c;
   %end;

   data &o;
   format ticker $8.;
   retain minwgt &minwgt maxwgt &maxwgt;
   %do j = 1 %to &_c0;
      ticker = "&&_c&j";
      output;
   %end;
   run;
   %if &i ne 1 %then %do;
      proc append base=&out_ds data=&o;
      run;
   %end;
   %else %do;
      %trash(_c);
   %end;

%mend;
```

```
/* return calculations
include $FPPCODE/pop/optimizer/retcalc.macro
depends on
$FPPCODE/utilities/macroman.macro
$FPPCODE/utilities/dsutil.macro
$FPPCODE/utilities/assets.macro
$FPPCODE/utilities/keyvar.macro
general dataset types:
ctw - (cticker,ticker,wgt) a collection of portfolios.  For example, cticker=fund tick
er, ticker=position ticker,wgt is wgt of ticker in fund
tv  - (ticker,var1,...) ticker-values.  presumed to be sorted or indexed by ticker, wh
ich should be unique.
dtt - (date,ticker,ticker...) return series
dtr - (date,ticker,return) elongated form of return series
"partition" is used in the set-theoritic sense: mutually exclusive and jointly exhaust
ive
ma refers to a macro array prefix, for example a1,a2,...  by convention, a0 contains t
he number of elements in a.
*/

%macro totalret(in_dtt=, in_dtr=, out_dtr=, out_dtt=,dropnull=0,zeronull=1);
 /* computes cumulative return from a return series
  parameters
  (choose in_dtt|in_dtr)
  in_dtt(date,ticker1,ticker2,...) return series
  in_dtr(date,ticker,return) return series
  (choose out_dtt | out_dtr)
  out_dtt(date,ticker1,ticker2,...) 1 row of total returns
  out_dtr(date,ticker,return) total returns
  zeronull =1 means set all null values to zero
  dropnull = 1 means remove all dates for which some return is null
   */
   %if &in_dtt&in_dtr eq or &out_dtt&out_dtr eq %then %do; %put totalret  parms missin
g; %goto exit; %end;

   %if &in_dtr ne %then %do;
      %cleanret(in_dtr=&in_dtr, out_dtt=_tr1,dropnull=&dropnull,zeronull=&zeronull);
   %end;
   %else %do;
      %cleanret(in_dtt=&in_dtt, out_dtt=_tr1,dropnull=&dropnull,zeronull=&zeronull);
   %end;
   %let in_dtt = _tr1;
   %if &out_dtt eq %then %do;
      %let out_dtt = _tr2;
   %end;

   %dsv2FL(in_ds=&in_dtt,mf=_f,ml=_l,cnt=_c,drop=date);


   data &out_dtt(drop = __x1 - __x&_c _i );
   set &in_dtt end=eof;
   array __t[&_c] &_f -- &_l;
   array __x[&_c];
   retain __x1 - __x&_c 1;
   do _i = 1 to &_c;
      __x[_i] = __x[_i] * (1+ __t[_i]);
      end;
   if eof then do;
      do _i = 1 to &_c;
         __t[_i] = __x[_i] - 1;
         end;
      output;
      end;
   run;
   %if &out_dtr ne %then %do;
      %w2ltran(in_ds=&out_dtt, out_ds=&out_dtr, by=date, id=ticker, var=return);
   %end;
   %trash(_tr1 _tr2);

%exit:
%mend;

%macro totret_t(in_dtt=, ticker=, outret=);
 /* computes cumulative return from a return series of a single ticker to a macro var
  parameters
  in_dtt(date, ticker1, ticker2, ....) return series
  ticker name of ticker to compute
  outret name of macro to output cumulative return to
  */

   %if &in_dtt eq or &outret eq or ticker eq %then %do; %put totret_t  parms missing;
%goto exit; %end;
   %global &outret;

   data _null_;
   set &in_dtt(keep=&ticker) end=eof;
   retain __x 1;
   __x = __x * (1+&ticker);
   if eof then do;
      __x = __x - 1;
      call symput("&outret", trim(left(put(__x, 15.8))));
      end;
   run;
%exit:

%mend;



%macro  runret(in_dtt=, in_dtr=, out_dtr=, out_dtt=,sort=1);
 /* produces a running total, i.e. cumulative,  return series
  parameters
  (choose in_dtt|in_dtr)
  in_dtt(date,ticker1,ticker2,...) return series
  in_dtr(date,ticker,return) return series
  (choose out_dtt | out_dtr)
  out_dtt(date,ticker1,ticker2,...) 1 row of total returns
  out_dtr(date,ticker,return) total returns
  sort - 1 means sort the output by date ticker
  */
   %if &in_dtt&in_dtr eq or &out_dtt&out_dtr eq %then %do; %put runret  parms missing;
 %goto exit; %end;


   %local i;
   %if &in_dtt&out_dtt eq %then %do;
      proc sort data=&in_dtr out=&out_dtr;
      by ticker date;
      data &out_dtr(keep=date ticker return);
      set &out_dtr(rename=(return=incret));
      by ticker date;
      retain return 0;
      if first.ticker then return=0;
      return = (1+incret) * (1+return) - 1;
      run;
      %if &sort ne %then %do;
         proc sort data=&out_dtr;
         by date ticker;
         run;
      %end;
```

```
   %end;
   %else %do;
      %cleanret(in_dtr=&in_dtr, in_dtt=&in_dtt, out_dtt=_rr1,zeronull=0);
      %if &out_dtt eq %then %do;
         %let out_dtt = _rr2;
      %end;
      %dsv2ma(in_ds=_rr1, drop=date, ma=_tk);

      data &out_dtt(keep = date %do i = 1 %to &_tk0; &&_tk&i %end;);
      set _rr1;
      array __x[&_tk0];
      array _r[&_tk0] %do i = 1 %to &_tk0; &&_tk&i %end;;
      retain __x1 - __x&_tk0 0;
      do _i = 1 to &_tk0;
         _r[_i] = (1+__x[_i]) * (1+_r[_i]) - 1;
         __x[_i] = _r[_i];
         end;
      run;
      %if &out_dtr ne %then %do;
         %w2ltran(in_ds=&out_dtt, out_ds=&out_dtr, by=date, id=ticker, var=return);
         %trash(_rr2);
      %end;
      %trash(_rr1);
   %end;

%exit:

%mend;


%macro rollret(in_dtt=, in_dtr=, roll=, out_dtt=, out_dtr=);
/* produces rolling return series
 i.e. each output date is the cumulative return of previous &roll dates of input retur
ns
  (choose in_dtt|in_dtr)
  in_dtt(date,ticker1,ticker2,...) return series
  in_dtr(date,ticker,return) return series
  roll - number of periods to roll by. if monthly returns and doing 3yr rolling, set r
oll=36
  (choose out_dtt | out_dtr)
  out_dtt(date,ticker1,ticker2,...) 1 row of total returns
  out_dtr(date,ticker,return) total returns
  */

   %local roll_p1 ;
   %let roll_p1 = %eval(&roll + 1);

   %runret(in_dtt=&in_dtt,in_dtr=&in_dtr,out_dtr=_rolldtr);
   proc sql;
   create table _rldt1 as
      select distinct date from _rolldtr order by date;
   data _rldt;
   merge _rldt1(rename=(date=pdate) in=_ina) _rldt1(firstobs= &roll_p1 in=_inb);
   if _inb;
   run;

   data _null_;
   set _rldt1(firstobs=&roll obs=&roll);
   call symput("_RR",trim(left(put(date,8.))));
   run;

   %if &out_dtr eq %then %let out_dtr  = _rrdtr;
   proc sql;
   create table &out_dtr as
      select a.date,a.ticker,(1+a.return)/(1+b.return) - 1 as return
      from _rolldtr a, _rolldtr b, _rldt c
      where a.date = c.date
      and b.date = c.pdate
      and a.ticker = b.ticker
   union
      select date,ticker,return
      from _rolldtr
      where date = &_rr
      order by date,ticker;

   %if &out_dtt ne %then %dtr2dtt(in_dtr=&out_dtr,out_dtt=&out_dtt);
   %trash(_rldt _rldt1 _rrdtr _rolldtr);

%mend;


%macro ctwret(in_ctw=, in_dtt=, in_dtr=, totaldtt=, totaldtr=, out_dtt=, out_dtr=, out
_ctw=, out_dctw=,dropnull=0,zeronull=1,rebal=0,missret=0);
 /* calculate portfolio returns
  parameters
  in_ctw(cticker,ticker,wgt)
  (choose in_dtt|in_dtr)
  in_dtt(date,ticker1,...) return series
  in_dtr(date,ticker,return) return series
  out_dtt(date,cticker1,...) optional return series for portfolios
  out_dtr(date,ticker,return) return series for portfolios
  totaldtt(date,cticker1,...) just the final returns
  totaldtr(date,ticker,return) just the final returns
  out_ctw(cticker,ticker,wgt) optional ending weights
  out_dctw(date,cticker,ticker,wgt) optional running weights
  dropnull - nonzero to indicate that dates with some null value should be excluded
             (use this if the returns are used for volatility purposes only)
  zeronull - zero means use null as null. otherwise convert null to zero
  rebal - nonzero to indicate that the portfolios are rebalanced to the initial weight
s
          each date.
  missret - nonzero to indicate that if an asset in ctw has no non-null returns,then r
emove it and renormalize ctw
  */

 /* _r2 = map of tickers to _w numbered tickid's
  _r3 = in_ctw with replaced tickers, widened
  _r5 = wide input returns, pruned and clean
  _r6 = renamed input returns
  _r7 = portfolio returns
  _r8 = final weights of portfolios
  */
   %local i;
   %if &in_ctw eq or &in_dtt&in_dtr eq %then %do; %put ctwret  parms missing; %goto ex
it; %end;
   %if &in_dtt ne %then %let in_dt = %quote(in_dtt=&in_dtt);
   %else %let in_dt = %quote(in_dtr=&in_dtr);
   data _r7;
   set &in_ctw(where=(wgt ne 0));
   run;
   %if &missret ne 0 %then %do;
      %if &in_dtt ne %then %do;
         %dtt2dtr(in_dtt=&in_dtt,out_dtr=_r5);
         %let i = _r5;
         %let in_dt = %quote(in_dtr=_r5);
      %end;
      %else %let i = &in_dtr;
```

```
      proc sql noprint;
      create table _hasret as
         select distinct ticker from &in_dtr where return ne .;
      create table _r7 as
         select a.cticker,a.ticker,a.wgt
         from _r7 a, _hasret b
         where a.ticker = b.ticker;
      %ctwnorm(in_ctw=_r7);
   %end;


   %if &out_dctw&out_dtt&out_dtr eq %then %do;

      /* no need for incremental results- just use total return. */

      %cleanret(%unquote(&in_dt),in_tv=_r7,out_dtt=_r5,zeronull=0,dropnull=&dropnull);

      %totalret(in_dtt=_r5,out_dtr=_r5,zeronull=&zeronull);
      %if &totaldtt&totaldtr ne %then %do;
         %if &totaldtr eq %then %let totaldtr = _r6;
         proc sql noprint;
         create table &totaldtr as
            select b.date, a.cticker as ticker, sum(a.wgt*(1+b.return)) -1 as return
            from _r7 a, _r5 b
            where a.ticker = b.ticker
            group by b.date, a.cticker
            order by b.date, a.cticker;
         %if &totaldtt ne %then %do;
            %dtr2dtt(in_dtr=&totaldtr,out_dtt=&totaldtt);
         %end;
      %end;
      %if &out_ctw ne %then %do;
         proc sql noprint;
         create table &out_ctw as
            select a.cticker, a.ticker, sum(a.wgt*(1+b.return)) as wgt
            from _r7 a, _r5 b
            where a.ticker = b.ticker
            group by  a.cticker,a.ticker
            order by  a.cticker,a.ticker;
         %ctwnorm(in_ctw=&out_ctw);
      %end;
   /*    %trash(_r5); */
    %end;
    %else %do;

      %varenum(in_ds=_r7,out_ds=_r2,prefix=_W);   /* convert tickers to numbered varia
bles */

      %ds2ma(in_ds=_r2,var=ticker,ma=_ct);
      %v_replac(in_ds=_r7,in_ref=_r2,var=ticker,repvar=tickid,out_ds=_r3);
      %l2wtran(in_ds=_r3,out_ds=_r3,by=cticker,id=ticker,var=wgt,sort=1);
      %dsnonull(in_ds=_r3);

      %if &in_dtt ne %then %do;
         %if &dropnull eq 0 %then %do;
            %cleanret(in_dtt=&in_dtt,in_tv=_r2,out_dtt=_r5,zeronull=&zeronull);
         %end;
         %else %do;
            %cleanret(in_dtt=&in_dtt,in_tv=_r2,out_dtt=_r5,zeronull=0);
            %cleanret(in_dtt=_r5,out_dtt=_r5,dropnull=1);
         %end;
      %end;
      %else %do;
         %if &dropnull eq 0 %then %do;
            %cleanret(in_dtr=&in_dtr,in_tv=_r2,out_dtt=_r5,zeronull=&zeronull);
         %end;
         %else %do;
            %cleanret(in_dtr=&in_dtr,in_tv=_r2,out_dtt=_r5,zeronull=0);
            %cleanret(in_dtt=_r5,out_dtt=_r5,dropnull=1);
         %end;
      %end;

      data _r6;
      set _r5(rename=(%do i = 1 %to &_ct0; &&_ct&i=_R&i %end; ));
      run;

      data _r7(keep=cticker date return)
         %if &out_ctw ne %then %do;
            _r8(keep=cticker _nw1 - _nw&_ct0
                rename=(%do i = 1 %to &_ct0; _nw&i=&&_ct&i %end; )) %end;
      %if &out_dctw ne %then %do;
         &out_dctw(keep= date cticker _nw1 - _nw&_ct0
                   rename=(%do i = 1 %to &_ct0; _nw&i=&&_ct&i %end; )) %end;;
      set _r3 nobs=_nobs;
      array _w[&_ct0] _w1 - _w&_ct0;
      array _nw[&_ct0];
      retain _nw1 - _nw&_ct0;
      retain _w1 - _w&_ct0 ;
      do _i = 1 to &_ct0;
         _nw[_i] = _w[_i];
         end;
      _zportf + 1;
      if _nobs > 50 and 10*int(_zportf/10) = _zportf then put "portfolio " _zportf " /
" _nobs;
      do _rno = 1 to _robs;
         set _r6 nobs=_robs point=_rno;
         array _r[&_ct0] _r1 - _r&_ct0;
         _sum = 0;
         return = 0;
         %if &rebal eq 0 %then %do;
            do _i = 1 to &_ct0;
               if _nw[_i] ne 0 then do;
                  return = return + _nw[_i] * _r[_i];
                  _nw[_i] = _nw[_i] * (1+_r[_i]);
                  _sum = _sum + _nw[_i];
               end;
            end;
            do _i = 1 to &_ct0;
               _nw[_i] = _nw[_i] / _sum;
            end;
         %end;
         %else %do;
            do _i = 1 to &_ct0;
               if _nw[_i] ne 0 then do;
                  return = return + _nw[_i] * _r[_i];
               end;
            end;
         %end;
         output _r7;
         %if &out_dctw ne %then output &out_dctw;;
      end;
      %if &out_ctw ne %then output _r8;;
      run;

      %if &out_ctw ne %then %do;
         %w2ltran(in_ds=_r8,out_ds=&out_ctw,by=cticker,id=ticker,var=wgt);
         data &out_ctw;
         set &out_ctw(where=(wgt ne 0));
         run;
      %end;
```

```
      %if &out_dctw ne %then %do;
         %w2ltran(in_ds=&out_dctw,out_ds=&out_dctw,by=date cticker,id=ticker,var=wgt,s
ort=1);
         data &out_dctw;
         set &out_dctw(where=(wgt ne 0));
      run;
      %end;
      %if &out_dtt&out_dtr ne %then %do;

         %if &out_dtt eq %then %let out_dtt=&out_dtr;
         proc sort data=_r7;
         by date cticker;

         proc transpose data=_r7 out=&out_dtt(keep = _numeric_);
         var return;
         id cticker;
         by date;
         run;
         %if &out_dtr ne %then %dtt2dtr(in_dtt=&out_dtt,out_dtr=&out_dtr);

      %end;

/*       %trash(_r2 _r3  _r5 _r6 _r7 %if &out_ctw ne %then _r8;); */
   %end;
%exit:

%mend;

%macro dctwret(in_dctw=, in_dtt=, in_dtr=,  out_dtt=, out_ctw=, out_dctw=,dropnull=0,z
eronull=1,missret=0);
/* calculates portfolio returns taking into account periodic rebalancings
assumes that when any portfolio gets rebalanced, they all do
  parameters
  in_dctw(date,cticker,ticker,wgt) weights as of beginning of each date
  (choose in_dtt|in_dtr)
  in_dtt(date,ticker1,...) return series
  in_dtr(date,ticker,return) return series
  out_dtt(date,cticker1,...) optional return series for portfolios
  out_ctw(cticker,ticker,wgt) optional ending weights
  out_dctw(date,cticker,ticker,wgt) optional running weights
  dropnull - nonzero to indicate that dates with some null value should be excluded
             (use this if the returns are used for volatility purposes only)
  zeronull - zero means use null as null. otherwise convert null to zero
  missret - nonzero to indicate that if an asset in ctw has no non-null returns,then r
emove it and renormalize ctw
  note that dates in &in_dctw and &in_dtr must be of the same form, i.e. either both Y
YYYMM or both YYYYMMDD
  (use %ym2eom on dctw if necessary)
*/
   %local d dp1 o_dtt o_ctw o_dctw _ctwend;
   %if &in_dtt eq %then %do;
      %let in_dtt=_indtt;
      %dtr2dtt(in_dtr=&in_dtr,out_dtt=_indtt);
   %end;

   %ds2ma(in_ds=&in_dctw,var=date,num=1,ma=_dds);
   proc sql noprint;
   select max(date) into :_maxdt
      from _indtt;
   %if %eval(&_maxdt > &&_dds&_dds0) %then %do;
      %ma_app(ma=_dds,value=&_maxdt);
   %end;
   %let _ctwend = %eval(&_dds0 - 1);
   %local dd;
   %let dd = 0;

   %do d = 1 %to &_ctwend;
      %let dp1 = %eval(&d + 1);
      data _ctw(drop=date);
      set &in_dctw(where=(date eq &&_dds&d));
      run;
      data _in_dtr;
      set &in_dtr(where=( date gt &&_dds&d and date le &&_dds&dp1));
      run;
      proc sql noprint;
      select count(*) into :_retcnt
         from _in_dtr;
      %if %eval(&_retcnt > 0) %then %do;

         %let dd = %eval(&dd+1);
         %if &dd eq 1 %then %do;
            %let o_dtt=&out_dtt;
            %let o_dctw=&out_dctw;
         %end;
         %else %do;
            %if &out_dtt ne %then %let o_dtt=_od1;
            %if &out_dctw ne %then %let o_dctw=_od3;
         %end;
         %ctwret(in_ctw=_ctw, in_dtr=_in_dtr,
                 out_dtt=&o_dtt, out_ctw=&out_ctw, out_dctw=&o_dctw,dropnull=&dropnull
,zeronull=&zeronull,missret=&missret);

         %if &dd ne 1 %then %do;
            %if &out_dtt ne %then %do;
               data &out_dtt;
               set &out_dtt &o_dtt;
               run;
            %end;
            %if &out_dctw ne %then %do;
               proc append base=&out_dctw data=&o_dctw;
               run;
            %end;
         %end;
      %end;
   %end;

%mend;


%macro ctwrevr(in_ctw=, in_dtt=, in_dtr=,  out_dtt=, out_dtr=, dropnull=0,zeronull=1,r
ebal=0,missret=0);
 /* reverse calculate ctwret
  parameters
  in_ctw(cticker,ticker,wgt)
  (choose in_dtt|in_dtr)
  in_dtt(date,ticker1,...) return series
  in_dtr(date,ticker,return) return series
  out_dtt(date,cticker1,...) optional return series for portfolios
  out_dtr(date,ticker,return) return series for portfolios
  dropnull - nonzero to indicate that dates with some null value should be excluded
             (use this if the returns are used for volatility purposes only)
  zeronull - zero means use null as null. otherwise convert null to zero
  missret - nonzero to indicate that if an asset in ctw has no non-null returns,then r
emove it and renormalize ctw
  rebal - nonzero to indicate that the portfolios are rebalanced to the initial weight
s
          each date.
```

```
   */
    %if &in_dtt eq %then %do;
       %dtr2dtt(in_dtr=&in_dtr,out_dtt=_revdtt);
       %let in_dtt=_revdtt;
    %end;

    data _revdtt(drop=_name);
    format date 10.;
    set &in_dtt;
    length _name $8;
    array __x _numeric_;
    do over __x;
       call vname(__x,_name);
       if _name eq 'DATE' then __x = 30000000 - __x;
       else __x = (1/(1+__x))-1;
       end;
    run;

    proc sort data=_revdtt;
    by date;
    run;
    %if &out_dtt eq %then %let out_dtt=_revout;
    %ctwret(in_ctw=&in_ctw, in_dtt=_revdtt,  out_dtt=&out_dtt, dropnull=&dropnull,zeron
ull=&zeronull,rebal=&rebal,missret=&missret);

    data &out_dtt(drop=_name);
    set &out_dtt;
    length _name $8;
    array __x _numeric_;
    do over __x;
       call vname(__x,_name);
       if _name eq 'DATE' then __x = 30000000 - __x;
       else __x = (1/(1+__x))-1;
       end;
    run;

    proc sort data=&out_dtt;
    by date;
    run;
    %if &out_dtr ne %then %dtt2dtr(in_dtt=&out_dtt,out_dtr=&out_dtr);
    ;

%mend;

%macro dctwrevr(in_dtr=,in_dctw=,out_dtt=,out_ctw=,out_dctw=,dropnull=0,zeronull=1,mis
sret=0);
/* reverse calculate dctwret
in this case, portfolio returns are calculated backwards.  For example, if in_dctw con
tains
end-of-month positions, and in_dtr is a daily return series of the consitituents, then
 the
portfolio returns for each month are calculated going backwards.
  parameters
  in_dctw(date,cticker,ticker,wgt) weights as of beginning of each date
  (choose in_dtt|in_dtr)
  in_dtr(date,ticker,return) return series
  out_dtt(date,cticker1,...) optional return series for portfolios
  out_ctw(cticker,ticker,wgt) optional ending weights
  out_dctw(date,cticker,ticker,wgt) optional running weights
  dropnull - nonzero to indicate that dates with some null value should be excluded
             (use this if the returns are used for volatility purposes only)
  zeronull - zero means use null as null. otherwise convert null to zero
  missret - nonzero to indicate that if an asset in ctw has no non-null returns,then r
emove it and renormalize ctw
  note that dates in &in_dctw and &in_dtr must be of the same form, i.e. either both Y
YYMM or both YYYYMMDD
  (use %ym2eom on dctw if necessary)
*/

  data _rev;
   format date 10.;
   set &in_dtr;
   return=(1/(1+return))-1;
   date = 30000000-date;
   run;
   proc sort data=_rev;
   by date;
   run;

   data _rdctw;
   format date 10.;
   set &in_dctw;
   date = 30000000-date -1;
   run;


   %dctwret(in_dctw=_rdctw,in_dtr=_rev,out_dtt=&out_dtt,out_ctw=&out_ctw,out_dctw=&out
_dctw,missret=&missret);

   %dsv2ma(in_ds=&out_dtt,drop=date,ma=_rev);

   data &out_dtt(drop=_name);
   set &out_dtt;
   length _name $8;
   array __x _numeric_;
   do over __x;
      call vname(__x,_name);
      if _name eq 'DATE' then __x = 30000000 - __x;
      else __x = (1/(1+__x))-1;
   end;
   run;

   proc sort data=&out_dtt;
   by date;
   run;

   %if &out_dctw ne %then %do;

      data &out_dctw;
      set &out_dctw;
      date=30000000 - date;
      run;

      proc sort data=&out_dctw;
      by date;
      run;
   %end;

%mend;


%macro cleanret(in_dtt=, in_dtr=, in_tv=, vars=, out_dtt=, out_dtr=,zeronull=1,dropnul
l=0);
 /* clean up a return series based on ticker list
  removes tickers not in in_tv,
  adds tickers with zero return where missing
  sets missing values to zero
  parameters
  (choose in_dtt|in_dtr)
  in_dtt(date,ticker1,...) return series
```

retcalc.macro


```
in_dtr(date,ticker,return) return series
in_tv(ticker,...) optional tickers to use
vars - list of tickers (instead of in_tv)
(choose out_dtt|out_dtr)
out_dtt(date,ticker1,...) defaults to in_dtt
out_dtr(date,ticker,return) alternative to out_dtt
zeronull =1 means set all null values to zero
dropnull = 1 means remove all dates for which some return is null
*/
 %if &in_dtt&in_dtr eq %then %do; %put cleanret  parms missing; %goto exit; %end;
 %if &out_dtt eq %then %do;
    %if &out_dtr eq %then %let out_dtt = &in_dtt;
    %else %let out_dtt = _rc3;
 %end;


 %if &in_tv ne %then %do;

    %if &in_dtr ne %then %do;
       %let in_dtt = _rc1;
       proc sql;
       create table &in_dtt as
          select distinct r.date,r.ticker,r.return
          from &in_tv t join &in_dtr r on r.ticker=t.ticker
          order by date;
       %l2wtran(in_ds=&in_dtt,by=date,var=return,id=ticker);
    %end;

    %global _CRA _CRZ;

    proc sql;
    create table _rc2 as
       select distinct ticker
       from &in_tv;

    data _rc2;
    set _rc2 end=eof;
    if _n_ eq 1 then call symput("_CRA",ticker);
    if eof then call symput("_CRZ",ticker);
    return = 0;
    run;
    %l2wtran(in_ds=_rc2,var=return,id=ticker);

    data &out_dtt(keep=date &_CRA -- &_CRZ);
    merge _rc2 &in_dtt;
    %if &dropnull eq 1 %then %do;
       array __x &_CRA -- &_CRZ;
       do over __x;
          if __x = . then return;
       end;
    %end;
    %else %if &zeronull eq 1 %then %do;
       array __x &_CRA -- &_CRZ;
       do over __x;
          if __x = .  then __x = 0;
       end;
    %end;
    output;
    run;

    %if &out_dtr ne %then %w2ltran(in_ds=&out_dtt,out_ds=&out_dtr,by=date,var=return
,id=ticker);
    %trash(%if &in_dtr ne %then _rc1; _rc2 %if &out_dtr ne %then _rc3;);
  %end;
  %else %do;
    %if &in_dtr ne %then %do;
       %let in_dtt = &out_dtt;
       %l2wtran(in_ds=&in_dtr,out_ds=&in_dtt,by=date,var=return,id=ticker);
    %end;
    data &out_dtt;
    set &in_dtt;
    %if &dropnull eq 1 %then %do;
       array __x _numeric_;
       do over __x;
          if __x = . then return;
       end;
    %end;
    %else %if &zeronull eq 1 %then %do;
       array __x _numeric_;
       do over __x;
          if  __x = . then __x = 0;
       end;
    %end;
    output;
    run;

    %if &out_dtr ne %then %w2ltran(in_ds=&out_dtt,out_ds=&out_dtr,by=date,var=return
,id=ticker);
    %trash( _rc3);
  %end;


%exit:

%mend;



%macro ret2tv(in_dtt=, in_tv=, out_tv=, var=return, total=0);
 /* puts total return into a var
  parameters
  in_dtt(date,ticker1,...) return series
  in_tv(ticker,var1,...) ac to attach return to, if any
  var - what to call the return
  out_tv(ticker,var1...) defaults to in_tv
  total - if nonzero, in_dtt has the total return already
  */
   %if &in_dtt eq %then %do; %put ret2tv  parms missing; %goto exit; %end;
   %if &out_tv eq %then %let out_tv = &in_tv;

   %if &total eq 0 %then %totalret(in_dtt=&in_dtt,out_dtt=_ret);;

   proc transpose data=%if &total eq 0 %then _ret; %else &in_dtt; (drop=date)
   out=_a4(rename=(col1=&var)) name=ticker;

   proc sort data=_a4;
   by ticker;

   %if &in_tv ne %then %do;
      data &out_tv;
      merge &in_tv(in=int) _a4(keep=ticker &var);
      by ticker;
      if int;
      run;
   %end;
   %else %do;
      data &out_tv;
      set _a4;
      run;
   %end;
```

```
   %trash(_a4);
%exit:

%mend;


%macro retcombo(in_dtt1=,in_dtt2=, out_dtt=, out_def=, tran=1);
 /* make return combo's of every ticker a-b for a in in_dtt1, b in in_dtt2
  parameters
  in_dtt1(date,ticker1,...)
  in_dtt2(date,ticker1,...)
  out_dtt(date,artificial-ticker1,...)
  out_def(ticker,ticker1,ticker2) the mapping rule back from out_dtt
  tran - transpose to wide format
  */

   %dsv2vds(in_ds=&in_dtt1,drop=date,out_ds=_r8);
   %dsv2ma(in_ds=&in_dtt1,drop=date,ma=_ra);
   %varenum(in_ds=_r8,out_ds=_r81,var=var);

   %dsv2vds(in_ds=&in_dtt2,drop=date,out_ds=_r9);
   %dsv2ma(in_ds=&in_dtt2,drop=date,ma=_rb);
   %varenum(in_ds=_r9,out_ds=_r91,var=var);
   %if &tran eq 0 %then %let o = &out_dtt; %else %let o = _r1;

   data  &o(keep=ticker date return) _r2(keep=ticker _ra _rb);
   merge &in_dtt1 &in_dtt2;
   by date;
   array __xa[&_ra0] &_ra1 -- &&_ra&_ra0;
   array __xb[&_rb0] &_rb1 -- &&_rb&_rb0;
   do _ra = 1 to &_ra0;
      do _rb = 1 to &_rb0;
         __x = (_ra-1)*&_rb0 + _rb;
         ticker = "CX"||left(trim(put(__x,8.)));
         if _n_ = 1 then output _r2;
         return = __xa[_ra] - __xb[_rb];
         output &o;
         end;
      end;
   run;

   proc sort data=&o;
   by date;

   %if &tran ne 0 %then %do;
      %l2wtran(in_ds=&o,out_ds=&out_dtt,var=return,by=date,id=ticker);
   %end;

   proc sql;
   create table &out_def as
      select _r2.ticker,_r81.var as ticker1,_r91.var as ticker2
      from _r2, _r81, _r91
      where _ra = _r81.tickid
      and   _rb = _r91.tickid
      order by ticker;

   %trash(_r1 _r2 _r8 _r9 _r81 _r91);
%exit:
%mend;


%macro deadrow(in_dtt=, out_dtt=);
 /* remove dead return rows
  */
   %if &out_dtt eq %then %let out_dtt=&in_dtt;
   data &out_dtt(drop=_nonmt);
   set &in_dtt;
   array __x _numeric_;
   _nonmt = 0;
   do over __x;
      if __x ne . and __x ne 0 then _nonmt = _nonmt + 1;
      if _nonmt > 1 then goto nonz;
      end;
   nonz: if _nonmt > 1 then output;
   run;
%exit:

%mend;

%macro inc_ret(in_dtt=, out_dtt=);
 /* find incomplete returns i.e. ones where there are missing values at start or end
  */
   data _bad(keep=ticker);
   set &in_dtt nobs = _nobs;
   format ticker $8.;
   array __x _numeric_;
   if _n_ eq 1 or _n_ = _nobs then
      do over __x;
         if __x eq . then do;
            call vname(__x,ticker);
            output;
            end;
         end;
   run;
   %ds2ma(in_ds=_bad,var=ticker,ma=_bad);
   data &out_dtt;
   set &in_dtt(keep=date %ma2l(ma=_bad));
   run;
%exit:

%mend;


%macro act_ret(in_dtt, benchmrk, out_dtt);
/* calculate active return
 parameters
 in_dtt(date,ticker,...) return series
 benchmrk - ticker of benchmark
*/

   %if &out_dtt eq %then %let out_dtt = &in_dtt;
   %let benchmrk = %upcase(&benchmrk);

   data &out_dtt(drop=&benchmrk);
   set &in_dtt;
   array _r _numeric_;
   do over _r;
      if "&benchmrk" ne vname(_r) then _r = _r - &benchmrk;
      end;
   run;
%exit:

%mend;

%macro dtt2dtr(in_dtt=, out_dtr=, sort=0);
/* convert (date,ticker1,ticker2...) to (date,ticker,return)
*/
   %w2ltran(in_ds=&in_dtt,out_ds=&out_dtr,by=date,id=ticker,var=return,sort=&sort);

%mend;
```

```
%macro dtr2dtt(in_dtr=, out_dtt=, sort=0);
/* convert (date,ticker,return) to (date,ticker1,ticker2...)
*/
   %l2wtran(in_ds=&in_dtr,out_ds=&out_dtt,by=date,id=ticker,var=return,sort=&sort);

   data &out_dtt;
   set &out_dtt(where=(date ne .));
   run;


%mend;

%macro tkrange(ticker=, from_dt=, to_dt=, in_dtt=, in_dtr=);
/* get live date range for a ticker in a return series
  parameters:
 ticker - name of ticker
 from_dt name of macro variable
 to_dt name of macro variable
 (choose in_dtt | in_dtr)
 in_dtt - return series
 in_dtr - return series
*/
%global &from_dt &to_dt;

 %if &in_dtt ne %then %do;
    proc sql noprint;
    select min(date),max(date)
       into :&from_dt, :&to_dt
       from &in_dtt
       where &ticker ne .;
   %end;
   %else %do;
     proc sql noprint;
    select min(date),max(date)
       into :&from_dt, :&to_dt
       from &in_dtt
       where ticker="&ticker" and return ne .;
   %end;


%mend;



%macro dtt_upto(in_dtt=,out_dtt=,date=,qty=,include=1);
/* get up to &qty returns up to &date
 include = 1 means include &date, 0 means not
*/
   %local q;
   %if &include eq 0 %then %let q = %eval(&qty + 1);
   %else %let q = &qty;

data _null_;
   set &in_dtt(keep=date);
   if date = &date then do;
      if _n_ < &q then _x=1;
      else _x = _n_ - &q + 1;
      call symput("_start",trim(left(put(_x,8.))));
      stop;
   end;
   if date > &date then do;
      if _n_ <= &qty then _x=1;
      else _x = _n_ - &qty;
      call symput("_start",trim(left(put(_x,8.))));
      stop;
   end;
   run;

data &out_dtt;
   set &in_dtt(where=(date %if &include eq 0 %then <; %else <=; &date));
   if _n_ >= &_start;
   run;

%mend;


%macro gpret(in_tv=, grpvar=, in_dtt=, in_dtr=, out_dtt=, out_dtr= );
/* Grouped returns.
in_tv(ticker,&grpvar) - group assignment
grpvar - should be $8. or smaller group names
choose(in_dtt | in_dtr)
in_dtt(date,ticker1,ticker2,...) return series
in_dtr(date,ticker,return) return series
out_dtt(date,ticker1,ticker2,...) output return series of each group
out_dtr(date,ticker,return) output return series of each group
avgfirst - 1 means take  returns of averages of returns by group,
           0 means take average by group of  returns
*/

   %if &out_dtt ne %then %let out_dt = %quote(out_dtt=&out_dtt);
   %else %let out_dt = %quote(out_dtr=&out_dtr);

   %if &in_dtt ne %then %do;
      %dtr2dtt(in_dtt=&in_dtt,out_dtr=_gpdtr);
      %let in_dtr=_grdtr;
   %end;
   %if &out_dtr eq %then %let out_dtr=_gpdtr;

   proc sql;
   create table &out_dtr as
      select a.date,b.&grpvar as ticker,avg(return) as return
      from &in_dtr a, &in_tv b
      where a.ticker=b.ticker
      group by a.date,b.&grpvar;
   %if &out_dtt ne %then %dtr2dtt(in_dtr=&out_dtr,out_dtt=&out_dtt);

   %trash(_gpdtr);


%mend;

%macro gpcumret(in_tv=, grpvar=, in_dtt=, in_dtr=, out_dtt=, out_dtr=, avgfirst=1 );
/* Grouped Cumulative returns.
in_tv(ticker,&grpvar) - group assignment
grpvar - should be $8. or smaller group names
choose(in_dtt | in_dtr)
in_dtt(date,ticker1,ticker2,...) return series
in_dtr(date,ticker,return) return series
out_dtt(date,ticker1,ticker2,...) output return series of each group
out_dtr(date,ticker,return) output return series of each group
avgfirst - 1 means take cumulative returns of averages of returns by group,
           0 means take average by group of cumulative returns
*/

   %if &out_dtt ne %then %let out_dt = %quote(out_dtt=&out_dtt);
   %else %let out_dt = %quote(out_dtr=&out_dtr);

   %if &avgfirst eq 1 %then %do;
      %if &in_dtt ne %then %do;
```

```
      %dtr2dtt(in_dtt=&in_dtt,out_dtr=_gpdtr);
      %let in_dtr=_grdtr;
   %end;

   proc sql;
   create table _gpdtr as
      select a.date,b.&grpvar as ticker,avg(return) as return
      from &in_dtr a, &in_tv b
      where a.ticker=b.ticker
      group by a.date,b.&grpvar;
   %if &out_dtt ne %then %runret(in_dtr=_gpdtr,out_dtt=&out_dtt);
   %else %runret(in_dtr=_gpdtr,out_dtr=&out_dtr);
%end;
%else %do;
   %if &in_dtt ne %then %runret(in_dtt=&in_dtt,out_dtr=_gpdtr);
   %else %runret(in_dtr=&in_dtr,out_dtr=_gpdtr);;

   %if &out_dtr eq %then %let out_dtr = _gpdtr;
   proc sql;
   create table &out_dtr as
      select a.date,b.&grpvar as ticker,avg(return) as return
      from _gpdtr a, &in_tv b
      where a.ticker=b.ticker
      group by a.date,b.&grpvar;
   %if &out_dtt ne %then %do;
      %dtr2dtt(in_dtr=&out_dtr,out_dtt=&out_dtt);
   %end;
%end;

%trash(_gpdtr);


%mend;

%macro gprolpct(in_tv=, grpvar=, in_dtt=, in_dtr=, out_ds=, pctiles=, roll= );
/* grouped rolling perctile returns
in_tv(ticker,&grpvar) - group assignment
grpvar - should be $8. or smaller group names
choose(in_dtt | in_dtr)
in_dtt(date,ticker1,ticker2,...) return series
in_dtr(date,ticker,return) return series
out_ds(date,ticker,pctile,return) - rolling percentile returns
pctiles - list of percentiles, eg 25 50 75
roll - number of periods to roll by. if monthly returns and doing 3yr rolling, set rol
l=36
*/


   %rollret(in_dtt=&in_dtt,in_dtr=&in_dtr, roll= &roll, out_dtr=_gpdtr);

   proc sql;
   create table _gpunv as
      select a.&grpvar, b.date,b.ticker,b.return
      from &in_tv a, _gpdtr b
      where a.ticker = b.ticker
      order by date, &grpvar;

   proc univariate data=_gpunv;
   var return;
   by date &grpvar;
   output out=&out_ds pctlpts=&pctiles pctlpre=_P;
   run;

   %trash(_gpdtr _gpunv);

%mend;


%macro ctw_hole(in_ctw=, in_dtr=, out_ctw=, out_ds= ,yes_ctw=, date=0);
/* find which returns are missing
in_ctw(cticker,ticker,wgt) portfolios for which returns are required
in_dtr(date,ticker,return) return series
out_ctw(cticker,ticker,wgt) subset of &in_ctw that don't have a full return series
out_ds(cticker,ticker,wgt,miscnt) ctw but including count of missing values
yes_ctw(cticker,ticker,wgt)optional ctw of those complete items
date - nonzero means in_ctw is (date,cticker,ticker,wgt)
*/

   %if &out_ctw eq %then %let out_ctw = &out_ds;

   proc sql noprint;
   create table _x as
      select ticker,count(*) as count
      from &in_dtr
      where return ne .
      group by ticker; /* nonnull tickers in &in_dtr */
   select max(count)
      into :maxcnt
      from _x;
   create table _y as
      select ticker
      from _x
      where count eq &maxcnt; /* good tickers */
   create table &out_ctw as
      select *
      from &in_ctw a
      where ticker not in (select ticker from _y)
      order by a.cticker, a.wgt desc;
   %if &yes_ctw ne %then %do;
      create table &yes_ctw as
         select a.*
         from &in_ctw a, _y
         where a.ticker=_y.ticker
         order by a.cticker,a.wgt desc;
   %end;

   %if &out_ds ne %then %do;
      create table _mx as
         select ticker,count
         from _x where ticker not in (select ticker from _y)
         union select distinct ticker, 0 as count
         from &in_ctw a
         where ticker not in (select ticker from _x);
      create table &out_ds as
         select %if &date ne 0 %then a.date,; cticker,a.ticker,a.wgt, &maxcnt - count
as miss_cnt
         from &out_ctw a,  _mx
         where a.ticker=_mx.ticker;
   %end;


%mend;

%macro ctwdmhol(in_ctw=, in_dtr=, out_ctw=, out_ds= ,yes_ctw=, months= );
/* find which returns are missing from monthly constituents and daily returns
in_ctw(cticker,ticker,wgt) portfolios for which returns are required
in_dtr(date,ticker,return) return series
out_ctw(cticker,ticker,wgt) subset of &in_ctw that don't have a full return series
```

```
out_ds(cticker,ticker,wgt,miscnt) ctw but including count of missing values
yes_ctw(cticker,ticker,wgt)optional ctw of those complete items
months -  this is the number of months forward (or if negative, backwards) to look for
 returns.
*/

   %ds2ma(in_ds=&in_ctw,var=date,num=1,ma=_dd);

   %local d yyyymm fromd tod;
   %do d = 1 %to &_dd0;
      %let yyyymm = &&_dd&d;
      %if %eval(&months > 0) %then %do;
         %dt_plusm(in_date=&yyyymm, months=1, out_date=fromm, dd=0);
         %let fromd = &fromm.01;
         %let tod   = &fromm.31;
      %end;
      %else %do;
         %let fromd = &yyyymm.01;
         %let tod   = &yyyymm.31;
      %end;

      %dsrange(in_ds=&in_ctw, out_ds=_in_ctw, var=date, from=&yyyymm, to=&yyyymm);

      %dsrange(in_ds=&in_dtr, out_ds=_in_dtr, var=date, from=&fromd, to=&tod);

      %ctw_hole(in_ctw=_in_ctw, in_dtr=_in_dtr, out_ctw=ctw_hole, out_ds=_out_ds ,yes_
ctw=_yes_ctw, date=1);

      %if &out_ctw ne %then  %dsappend(i=&d,in_ds=ctw_hole,out_ds=&out_ctw,same=1);;
      %if &out_ds  ne %then  %dsappend(i=&d,in_ds=_out_ds ,out_ds=&out_ds ,same=1);;
      %if &yes_ctw ne %then  %dsappend(i=&d,in_ds=_yes_ctw,out_ds=&yes_ctw,same=1);;

   %end;

%mend;


%macro dtr_hole(in_tv=, in_dtr=, out_ds= );
/* find which returns are missing
in_tv(ticker,...) tickers there should be returns for
in_dtr(date,ticker,return) return series
out_ds(date,ticker) date,ticker pairs that are missing
*/
   %ds2ma(in_ds=&in_tv,var=ticker,ma=rt);

   %local i;

   proc sql;
   create table &out_ds as
      select distinct b.date,a.ticker,b.return
      from &in_tv a left join &in_dtr b on a.ticker=b.ticker
      order by date,ticker;

   %dtr2dtt(in_dtr=&out_ds,out_dtt=&out_ds);

   data &out_ds(keep=date ticker);
   format ticker $8.;
   set &out_ds(where=(date ne .));
   array __x %do i = 1 %to &rt0; &&rt&i %end;;
   do over __x;
      if __x eq . then do;
         call vname(__x,ticker);
         output;
      end;
   end;
   run;

   proc sort data=&out_ds;
   by ticker date;
   run;


%mend;


%macro holiday0(in_dtt=, in_dtr=, out_dtr=, out_dtt=, reftick=_CASH, ref_dtt=);
/* zero holidays in a return series
reftick is the ticker of the reference series presumed to have
non-null returns for every non-holiday
*/

   %if &in_dtt eq %then %do;
      %dtr2dtt(in_dtr=&in_dtr,out_dtt=_noh);
      %let in_dtt = _noh;
   %end;
   %if &out_dtt eq %then %let out_dtt=_noh;

   data &out_dtt(drop = _name_);
   format _name_ $8.;
   merge &in_dtt(in=_ina) %if &ref_dtt ne %then &ref_dtt(keep=date &reftick);;
   by date;
   if _ina;
   array __x _numeric_;
   if &reftick eq . then do;
      do over __x;
         call vname(__x,_name_);
         if (_name_ not in("DATE")) then __x = 0;
      end;
   end;
   run;

   %if &out_dtr ne %then %do;
      %dtt2dtr(in_dtt=&out_dtt,out_dtr=&out_dtr);
   %end;

%mend;

%macro holidayX(in_dtt=, in_dtr=, out_dtr=, out_dtt=, reftick=_CASH, ref_dtt=);
/* remove holidays from a return series
rolls nonzero holiday returns (presumably from foreign securities) into next days retu
rns
*/
   %if &in_dtt eq %then %do;
      %dtr2dtt(in_dtr=&in_dtr,out_dtt=_noh);
      %let in_dtt = _noh;
   %end;
   %if &out_dtt eq %then %let out_dtt=_noh;

   %if &ref_dtt eq %then %let ref_dtt=&in_dtt;
   data _nohd(keep=date);
   set &ref_dtt(keep=date &reftick where=(&reftick ne .));
   run;

   %dsv2FL(in_ds=&in_dtt,drop=DATE,mf=_hf,ml=_ht,cnt=_hc);

   data &out_dtt(drop=_i __x1 - __x&_hc);
   format date 8. _i 10.2;
   merge  &in_dtt(in=_inr) _nohd(in=_ind) ;
   by date;
   array __t[&_hc] &_hf -- &_ht;
```

```
   array __x[&_hc]  ;
   retain __x1 - __x&_hc 0;
   do  _i = 1 to &_hc;
      if not _ind and __t[_i] eq . then __t[_i] = 0;
      __x[_i] = (__x[_i] + 1) * (1 + __t[_i]) - 1;
   end;
   if _ind then do;
      do _i = 1 to &_hc;
         __t[_i] = __x[_i];
         __x[_i] = 0;
      end;
      output;
   end;
   run;

   %if &out_dtr ne %then %do;
      %dtt2dtr(in_dtt=&out_dtt,out_dtr=&out_dtr);
   %end;

%mend;


%macro perioret(in_dtr=, in_dtt=, out_dtr=, out_dtt=, dateds=,dropnull=0,zeronull=1);
/* accumulates returns up to dates in dateds or to month ends
choose in_dtr | in_dtt
in_dtr(date,ticker,return) return series
in_dtt(date,ticker,ticker,...) return series
out_dtr(date,ticker,return) output return series
out_dtt(date,ticker,ticker,...) output return series
dateds(date) optional list of dates to accumulate to
dropnull - 1 means drop all dates where some return is missing
zeronull - 1 means interpret any missing returns as 0
if &dateds is not given, returns are aggregated to month-end
*/
   %if &in_dtt&in_dtr eq or &out_dtt&out_dtr eq %then %do; %put perioret  parms missin
g; %goto exit; %end;

   %if &in_dtr ne %then %do;
      %cleanret(in_dtr=&in_dtr, out_dtt=_tr1,dropnull=&dropnull,zeronull=&zeronull);
   %end;
   %else %do;
      %cleanret(in_dtt=&in_dtt, out_dtt=_tr1,dropnull=&dropnull,zeronull=&zeronull);
   %end;
   %let in_dtt = _tr1;
   %if &out_dtr ne %then %do;
      %let out_dtt = _tr2;
   %end;

   %if &dateds eq %then %do;
      proc sql;
      create table _dds as
         select distinct int(date/100) as date
         from _tr1;
      data _dds;
      set _dds;
      date=%df_eom(in_date=100*date+1);
      run;
   %end;
   %else %do;
      proc sql;
      create table _dds as
         select distinct date
         from &dateds
         order by date;
   %end;

   %dsv2FL(in_ds=&in_dtt,mf=pmf,ml=pml,cnt=pcnt,drop=date);

   data &out_dtt(drop =_i __x1 - __x&pcnt);
   format date 8. _i 10.2;
   merge  &in_dtt(in=_inr) _dds(in=_ind) ;
   by date;
   array __t[&pcnt] &pmf -- &pml;
   array __x[&pcnt]  ;
   retain __x1 - __x&pcnt 1;
   do _i = 1 to &pcnt;
      if not _inr then __t[_i] = 0;
      __x[_i] = __x[_i] * (1+ __t[_i]);
      end;
   if _ind then do;
      do _i = 1 to &pcnt;
         __t[_i] = __x[_i] - 1;
         __x[_i] = 1;
      end;
      output;
   end;
   run;

   %if &out_dtr ne %then %do;
      %w2ltran(in_ds=_tr2, out_ds=&out_dtr, by=date, id=ticker, var=return);
      %trash(_tr2);
   %end;
   %trash(_tr1);

%exit:
%mend;
```

```
%macro cov2ma(covmat=, ma=);
/* covariance matrix to macro array
macro array output is 123*ticker1*ticker2
for each pair ticker1,ticker2 where 123 is the covariance
used in constructing nonlinear constraint that portfolio variance equals a fixed numbe
r
*/

   proc transpose data=&covmat(rename=(_name_=_name2) where=(_type_='COV')) out=_cvt(d
rop=_type_);
   by _TYPE_ _NAME2;
   run;

   %global &ma.0;

   proc sql noprint;
   select count(*) into :&ma.0 from _cvt;

   %local i;

   %do i = 1 %to &&&ma.0;
      %global &&ma.&i;
   %end;

   data _null_;
   set _cvt ;
   ma=compress("&ma"||put(_n_,3.));
   call symput(ma,compress(put(col1,18.10)||'*'||_name_||'*'||_name2));
   run;

%mend;



/* taradj -
 adjust target asset allocations so that a portfolio of
 the given level of company stock plus the new target weights for indexes
 has the same volatility as that of the original weights without company stock.
 Uses NLP with the variance as a non-linear constraint and the return as the objective
 function to maximize.
*/


data ast_idx;
format var $8. ticker $8.;
input var $ ticker $ stocks dstocks fstocks bonds cash hybonds igbonds;
cards;
IGBONDS _BOND 0 0 0 1 0 0 1
CASH _CASH_ 0 0 0 0 1 0 0
DSTOCKS _W5000 1 1 0 0 0 0 0
FSTOCKS _EAFE 1 0 1 0 0 0 0
HYBONDS _ML_HYM 0 0 0 0 0 1 0
;

proc sql noprint;
   create index var    on ast_idx(var);
   create index ticker on ast_idx(ticker);


%macro var2ticker(in_ds=,out_ds=);
/* var to ticker
*/

   %if &out_ds eq %then %let out_ds=&in_ds;
   proc sort data=&in_ds out=&out_ds;
   by var;
   data &out_ds;
   length ticker $8;
   merge &out_ds(in=ina) ast_idx(keep=var ticker in=inb);
   by var;
   if ina and inb;
   run;

%mend;


%macro taradj(inconst=, in_dtt=,  fixed=, in_tv=, outconst=);
/* adjust target asset allocation based on fixed assets (eg company stock )
 inconst(var,target,range,minwgt,maxwgt) asset allocation constraints
 in_dtt(date,ticker,...) return series including fixed assets and indexes. this gets t
he _bench return added
 fixed(ticker,wgt) - list of assets with their fixed weights
 in_tv(ticker,dstocks,fstocks,cash,bond,...) asset allocations of fixed assets
 outconst(var,target,range,minwgt,maxwgt) asset allocation constraints
*/



   %global status;

   /* put original one in there in case... */

   data &outconst;
   set &inconst;
   minwgt=target-range;
   maxwgt=target+range;
   run;

   /* get list of tickers to be used for optimizations */

   proc sql noprint;
   select 1-sum(wgt) into :newmaxwt
      from &fixed;

   %if %sysevalf(&newmaxwt ne 1 and &newmaxwt ne .) %then %do;

      /* turn total stock target into domestic stock target */

      proc transpose data=&inconst(keep=var target) out=targets(drop=_label_);
      var target;
      id var;
      run;

      data targets;
      format other 8.4;
      set targets;
      DSTOCKS=STOCKS-FSTOCKS;
      IGBONDS=BONDS-HYBONDS;
      call symput("_EQTOT",compress(put(stocks*1.05,10.6)));
      run;

      proc transpose data=targets(drop=STOCKS BONDS OTHER) out=targets(rename=(_name_=
var));
      run;

      /* set up ranges based on original targets (don't want to stray too far from ori
ginal weights) */
```

```
      %var2ticker(in_ds=targets);

/*
      data aarange;
      set targets(keep=var ticker target) end=eof;
      if ticker eq "_EAFE" then maxwgt = target*1.2;
      else do;
         minwgt = 0;
         maxwgt = 1;
      end;
      output;
      if eof then do;
         var='STOCKS';
         ticker = '';
         minwgt = 0;
         maxwgt = 1;
         output;
      end;
      run;
*/
     data aarange;
      set targets(keep=var ticker target) end=eof;
      if ticker eq '_BOND' then minwgt=target*.75;
      else if ticker eq '_CASH_' then minwgt = target/2;
      if ticker eq '_W5000' then maxwgt = target*1.2 + &stkwgt1/100; /* assume only 1
domestic stock and no foreign */
      else if ticker eq "_EAFE" then maxwgt = target*1.2;
      else if ticker eq "_ML_HYM" then maxwgt = .1;
      if minwgt eq . then minwgt = 0;
      if maxwgt eq . then maxwgt = 1;
      output;
      if eof then do;
         var='STOCKS';
         ticker = '';
         minwgt = 0;
         maxwgt = &_EQTOT;
         output;
         var='BONDS';
         ticker = '';
         minwgt = 0;
         maxwgt = 1;
         output;
      end;
      run;

      proc sort data=aarange;
      by ticker;
      run;

      /* set up tv for optimization */

      data quad_tv(keep=ticker stocks dstocks fstocks igbonds bonds cash hybonds);
      merge aarange(in=ina where=(ticker ne '')) &fixed(in=inb) &in_tv(in=inc keep=tic
ker stocks dstocks fstocks bonds igbonds cash hybonds) ast_idx(in=ind);
      by ticker;
      if inb or ind;
      run;

      %ds2l(in_ds=quad_tv,list=_ticks,var=ticker);

      %put TARADJ: optimizing &_ticks;

      data _tadtt;
      merge &in_dtt( keep = date &_ticks) pf.mdl_dtt(keep=date _bench _eqbench );
      by date;
      run;


   /* get covariance matrix */


      data _cvin(drop=_bench);
      set _tadtt(keep=&_ticks _bench);
      array _x &_ticks;
      do over _x;
         _x = (_x-_bench)*100;
      end;
      run;

      proc corr cov data=_cvin outp=_cov nosimple noprint;
      var &_ticks;
      run;


/*
      proc print data=_cov;
      title 'taradj covariance matrix';
      run;
*/

     /* make asset of original target portfolio */

      %quad(in_tv=quad_tv, covmat=_cov, lambda=1, tconst=aarange, sconst=&fixed, inqua
d=pf.tarquad,outquad=pf.taroutquad,title=Minimum Tracking Error Portfolio);
      %if "&objfnval" ne "." %then %do;
         %q2ctw(outquad=pf.taroutquad,out_ctw=newtar,cticker=_newtar);
         %terr_ctw(in_ctw=newtar,bench=,in_dtt=_tadtt,out_ds=,outmv=tarterr,peryear=12
,out_dtt=,rebal=1);
         %put Adjusted Target Tracking Error = &tarterr;
         %if %sysevalf(&tarterr gt .25 * &TAR_STOCKS) %then %do;
            %put TARADJ: Minimum Tracking Error portfolio has tracking error of &tarte
rr;
            %goto giveup;
         %end;
      %end;
      %else %do;
         %put TARADJ: could not find Minimum Tracking Error portfolio subject to const
raints;
         %GOTO GIVEUP;
      %end;
      proc sql noprint;
      create table _newtar as
         select sum(wgt*(dstocks+fstocks)) as stocks, sum(wgt*fstocks) as fstocks, sum
(wgt*(igbonds+hybonds)) as bonds,
         sum(wgt*cash) as cash
         from quad_tv a, newtar b
         where a.ticker=b.ticker;


         %w2ltran(in_ds=_newtar,out_ds=_newtar,var=newtar,id=var);
         proc sort data=_newtar;
         by var;
         run;

         proc sort data=&inconst;
         by var;
         data &outconst(drop=newtar);
         merge _newtar(in=inb) &inconst;
         by var;
         if newtar ne . then do;
```

```
            target=newtar;
            end;
         minwgt=target-range;
         maxwgt=target+range;
         run;
 /*       %trash(_outcon _tadtt tacumret _cov _target quad_tv fconst tar_dtt maxvar mi
nvar newtar); */
   %end;

   %let status = 0;

   %if 1=2 %then %do;
   %GIVEUP:
      %let status = 1;
   %end;
%mend;


%macro covmat2var(in_cov=,in_ctw=,out_ds=,ma=);
/* calculate variance given covariance matrix and asset weights
in_cov - covariance matrix as produced by proc corr
in_ctw(cticker,ticker,wgt) asset weights for one or more portfolios
out_ds(cticker,variance) optional output variances
ma - optional name of macro that will contain variance (assumes there is only one port
folio in &in_ctw)
*/
   %if &ma ne %then %global &ma;

   data _c(drop=_type_ rename=(_name_=ticker));
   set &in_cov(where=(_type_ = 'COV'));
   run;

   %w2ltran(in_ds=_c,by=ticker,id=ticker1,var=cov,sort=1);


   %if &out_ds ne %then %do;

      %ds2ma(in_ds=&in_ctw,var=cticker,ma=ct);
      %local i j ip99 ip1;

      /* we break this up into sets of 1000 else it runs out of space */

      %do i = 1 %to &ct0 %by 1000;

         %let ip1 = %eval(&i + 1);
         %let ip99 = %eval(&i + 999);
         %if %eval(&ip99 > &ct0) %then %let ip99 = &ct0;

         %put covmat2var &i / &ct0;
         data _ctw;
         set &in_ctw(where=(cticker in ("&&ct&i" %do j = &ip1 %to &ip99; ,"&&ct&j" %en
d;
                                    )));
         run;

         proc sql;
         create table _var as
            select a.cticker,sum(a.wgt*b.wgt*c.cov) as variance
            from _ctw a, _ctw b, _c c
            where a.ticker = c.ticker
            and b.ticker = c.ticker1
            and a.cticker=b.cticker
            group by a.cticker;

         %dsappend(in_ds=_var,out_ds=&out_ds,i=&i);
      %end;

   %end;

   %if &ma ne %then %do;
      proc sql noprint;
      select sum(a.wgt*b.wgt*c.cov) as variance
         into :&ma
         from &in_ctw a, &in_ctw b, _c c
         where a.ticker = c.ticker
         and b.ticker = c.ticker1
         and a.cticker=b.cticker;
   %end;

%mend;
```

```
/* asset dataset manipulation
%include $FPPCODE/utilities/macros.macro

general dataset types:
ctw - (cticker,ticker,wgt) a collection of portfolios.  For example, cticker=fund tick
er, ticker=position ticker,wgt is wgt of ticker in fund
tv  - (ticker,var1,...) ticker-values.  presumed to be sorted or indexed by ticker, wh
ich should be unique.
dtt - (date,ticker,ticker...) return series
dtr - (date,ticker,return) elongated form of return series
A "partition" is a set of variables in a dataset whose values ideally sum to 1.
For example, in a dataset of fund characteristics, the sector weights form a partition
.
ma refers to a macro array prefix, for example a1,a2,...  by convention, a0 contains t
he number of elements in a.
*/

%macro ctwnorm(in_ctw=,  out_ctw=, default=, date=0);
 /* normalize a cticker-ticker-wgt set
Ensure sum of weights by cticker all add to 1.
If &default is present, keep existing values unchanged, and put difference in ticker =
 &default.
If &default is blank, proportionally adjust each existing weight so the sums add to 1.
  parameters
  in_ctw(cticker,ticker,wgt)
  out_ctw(cticker,ticker,wgt) - optional-defaults to in_ctw
  default - optional ticker name for part not adding to 1
date - nonzero means in_ctw is (date,cticker,ticker,wgt)

Example 1:
  data x;
  format cticker ticker $8.;
  input cticker ticker wgt;
 cards;
FMAGX T .4
FMAGX XOM .4
FCNTX T .2
FCNTX R .6
;
%ctwnorm(in_ctw=x);
 proc print data=x; run;
OBS    CTICKER     TICKER      WGT
 1     FCNTX       R          0.75
 2     FCNTX       T          0.25
 3     FMAGX       T          0.50
 4     FMAGX       XOM        0.50

Example 2:
  data x;
  format cticker ticker $8.;
  input cticker ticker wgt;
 cards;
FMAGX T .4
FMAGX XOM .4
FCNTX T .2
FCNTX R .6
;
%ctwnorm(in_ctw=x,default=_OTHER);
 proc print data=x; run;
OBS    CTICKER     TICKER      WGT
 1     FCNTX       R          0.6
 2     FCNTX       T          0.2
 3     FCNTX       _OTHER     0.2
 4     FMAGX       T          0.4
 5     FMAGX       XOM        0.4
 6     FMAGX       _OTHER     0.2
*/
  %if &date eq 0 %then %do;

    %kkvnorm(in_ds=&in_ctw,out_ds=&out_ctw,default=&default,
          vars=wgt,key1=cticker,key2=ticker);
  %end;
  %else %do;

    %kkvnorm(in_ds=&in_ctw,out_ds=&out_ctw,default=&default,
          vars=wgt,key2=cticker,key3=ticker,key1=date);
  %end;

%mend;



%macro ctw2tv(in_ctw=, out_tv=);
 /* converts a cticker-ticker-wgt to tv
Use this to look at which portfolios each ticker is in
  parameters
  in_ctw(cticker,ticker,wgt)
  out_tv(ticker,cticker1,cticker2,...) where values are weights

Example:
  data x;
  format cticker ticker $8.;
  input cticker ticker wgt;
 cards;
FMAGX T .5
FMAGX XOM .5
FCNTX T .3
FCNTX R .7
;
%ctw2tv(in_ctw=x, out_tv=y);
proc print data=y; run;
 OBS    TICKER      FCNTX     FMAGX
 1      R           0.7       0.0
 2      T           0.3       0.5
 3      XOM         0.0       0.5
*/

   %local _ctc;
   proc sort data=&in_ctw out=_ctw;
   by ticker;

   %l2wtran(in_ds=_ctw,out_ds=&out_tv,var=wgt,by=ticker,id=cticker);
   %dsnonull(in_ds=&out_tv);
   %trash(_ctw);

%mend;



%macro ctw_agg(in_ctw=, in_tv=, twgt=, cwgt=, out_ctw=, cticker=AGGREGAT);
 /* aggregate portfolios into a single portfolio
  parameters
  in_ctw(cticker,ticker,wgt) portfolios to be aggregated
  in_tv(ticker,&wgt) optional tv containing weighting variable for portfolios
  twgt - name of weighting variable in &in_tv for weighting tickers
  cwgt - name of weighting variable in &in_tv for weighting ctickers
  out_ctw(cticker,ticker,wgt) aggregate portfolio

Example:
```

```
  data x;
  format cticker ticker $8.;
  input cticker ticker wgt;
 cards;
FMAGX T .5
FMAGX XOM .5
FCNTX T .25
FCNTX R .75
;
%ctw_agg(in_ctw=x, out_ctw=z);
proc print data=z; run;
OBS    CTICKER     TICKER     WGT
 1     AGGREGAT     R        0.375
 2     AGGREGAT     T        0.375
 3     AGGREGAT     XOM      0.250
*/
   %if &twgt.&cwgt eq %then %do;

      proc sql noprint;
      create table &out_ctw(cticker char(8), ticker char(8), wgt num);
      insert into &out_ctw
         select "&cticker" ,ticker,sum(wgt)
         from &in_ctw
         group by "&cticker" ,ticker;
   %end;
   %else %do;
      proc sql noprint;
      create table &out_ctw(cticker char(8), ticker char(8), wgt num);
      insert into &out_ctw
         select "&cticker" , p.ticker,
         sum(p.wgt %if &twgt ne %then * t.&twgt; %if &cwgt ne %then * c.&cwgt; )
         from &in_ctw p %if &twgt ne %then , &in_tv t ; %if &cwgt ne %then , &in_tv c
            where 1=1
            %if &twgt ne %then and p.ticker = t.ticker;
      %if &cwgt ne %then and p.cticker = c.ticker;
      group by "&cticker" ,p.ticker;
   %end;
   %ctwnorm(in_ctw=&out_ctw);

%mend;

%macro ctwOPctw(in_ctw1=, in_ctw2=, out_ctw=);
 /* ticker1/ticker2 o ticker2/ticker3 -> ticker1/ticker3
  look through one ctw to the underlying assets
  if ctw1 are portfolios of portfolios in ctw2,
  then output the implied portfolios

Example:
  For Freedom Funds underlying holdings,
  %m_fndcon(from_dt=200101, tickers=FFFCX FFFDX FFFEX, out_ctw=ff_ctw); Freedom fund c
onstituent funds
  %m_fndcon(from_dt=200101,in_tv=ff_ctw,out_ctw=fnd_ctw);  constituent fund holdings
  %ctwOPctw(in_ctw1=ff_ctw, in_ctw2=fnd_ctw, out_ctw=fu_ctw); Freedom fund underlying
holdings
*/
   proc sql;
   create table &out_ctw as
      select a.cticker,b.ticker,sum(a.wgt*b.wgt) as wgt
      from &in_ctw1 a, &in_ctw2 b
      where a.ticker=b.cticker
      group by a.cticker,b.ticker;
   %ctwnorm(in_ctw=&out_ctw);

%mend;
```

```
%macro ctw_dupo(in_ctw=, out_ctw=, date=0);
 /* consolidates duplicates in cticker-ticker-wgt set
  also renames blank tickers to _UNKNOWN
  and removes records where wgt is 0
  parameters
  in_ctw(ticker,wgt)
  out_ctw(ticker,wgt) defaults to in_ctw
  date - nonzero means in_ctw is (date,cticker,ticker,wgt)

Example:
  data x;
  format cticker ticker $8.;
  input cticker ticker wgt;
 cards;
FMAGX T .3
FMAGX T .1
FMAGX XOM .4
FCNTX T .2
FCNTX R .6
FCNTX XOM 0
;
%ctw_dupo(in_ctw=x);
proc print data=x; run;
OBS    CTICKER     TICKER      WGT
 1     FCNTX       R           0.6
 2     FCNTX       T           0.2
 3     FMAGX       T           0.4
 4     FMAGX       XOM         0.4
*/
   %if &out_ctw eq %then %let out_ctw = &in_ctw;

   data &out_ctw;
   set &in_ctw(where=(wgt ne 0));
   if ticker eq '' then ticker= '_UNKNOWN';
   run;

   proc sql;
   create table &out_ctw as
      select %if &date  %then date,; cticker, ticker,sum(wgt) as wgt
      from &out_ctw
      group by %if &date  %then date,; cticker, ticker;


%mend;



%macro ctw_filt(in_ctw=, in_tv=, out_ctw=, filter=, sort=0);
 /* general ctw filter
  parameters
  in_ctw(cticker,ticker,wgt)
  in_tv(ticker,...) - ticker for this = cticker for in_ctw
  out_ctw(cticker,ticker,wgt) only those from in_ctw satistying filter
  filter - SAS where clause for in_tv
  sort - nonzero indicates in_ctw and in_tv need to be sorted
  data x;
  format cticker ticker $8.;
  input cticker ticker wgt;
 cards;
FMAGX T .5
FMAGX XOM .5
FCNTX T .3
FCNTX R .7
;
  data t;
```

```
  format ticker $8. indu $8.;
input ticker $ indu $;
  cards;
R
T TELCOM

 proc print data=x; run;

  */
   %if &sort ne 0 %then %do;
      proc sort data=&in_ctw;
      by cticker;
      proc sort data=&in_tv;
      by ticker;
   %end;
   data &out_ctw(keep=cticker ticker wgt);
   merge &in_ctw(in=ina) &in_tv(rename=(ticker=cticker) where=(&filter) in=inb);
   by cticker;
   if ina and inb;
   run;

%mend;



%macro ctwDVprt(in_ctw=, in_tv=, part=, out_ctw=,other=_OTHER);
 /* (cticker,ticker,WGT) DV (ticker,V1,...Vn) -> (cticker,Vi,WGTi)
  for example, convert from (fund,ticker,wgt) to (fund,sector,wgt)
  parameters
  in_ctw(cticker,ticker,wgt)
  in_tv(ticker,partvar1,partvar2,...)
  part - list of parition variables
  out_ctw(cticker,ticker,wgt) where ticker values are from &part and
  wgts are calculated from original asset weights and their parition weights
  */
   %vvnorm(in_ds=&in_tv,vars=&part,out_ds=_ap1,default=&other);
   proc sort data=&in_ctw out=_ap2;
   by ticker;
   run;
   data &out_ctw(keep=cticker ticker wgt);
   merge _ap1(keep=ticker &part) _ap2(in=ina rename=(wgt=_inwgt));
   by ticker;
   if ina;
   array _x &part;
   if _inwgt = . then _inwgt = 0;
   _totwgt = 0;
   do over _x;
      call vname(_x,ticker);
      if _x = . then _x = 0;
      wgt = _x * _inwgt;
      _totwgt = _totwgt + wgt;
      if wgt ne 0 then output;
   end;
   if "&OTHER" ne "" and _totwgt eq 0 and _inwgt ne 0 then do;
      ticker = "&OTHER";
      wgt = _inwgt;
      output;
   end;
   run;

   %ctw_dupo(in_ctw=&out_ctw);
   %ctwnorm(in_ctw=&out_ctw);

%mend;
```

```
%macro ctw_x(in_ctw1=, in_ctw2=, out_ctw=, out_def=, norm=1);
 /* all pairs of two ctw's to a new ctw
  parameters
  in_ctw1(cticker,ticker,wgt)
  in_ctw2(cticker,ticker,wgt)
  out_ctw(cticker,ticker,wgt)
  out_def(cticker,cticker1,cticker2)
  norm - normalize results
  */
   proc sql;
   create table _a1 as
      select distinct cticker as cticker1
      from &in_ctw1;
   create table _a2 as
      select distinct cticker as cticker2
      from &in_ctw2;
   create table &out_def as
      select cticker1, cticker2
      from _a1, _a2;

   data &out_def;
   set &out_def;
   format cticker $8.;
   cticker = "T_"||left(trim(put(_n_,6.)));
   run;

   proc sort data=&out_def;
   by cticker;
   run;

   proc sql;
   create table &out_ctw(cticker char(8), ticker char(8), wgt num);
   insert into &out_ctw
      select m.cticker,a.ticker,a.wgt * b.wgt as wgt
      from &out_def m, &in_ctw1 a, &in_ctw2 b
      where m.cticker1 = a.cticker
      and m.cticker2 = b.cticker
      and a.ticker = b.ticker;

   %if &norm ne 0 %then %ctwnorm(in_ctw=&out_ctw);

%mend;

%macro ctwV2ctw(in_ctw=, in_tv=,var=, out_ctw=, out_tv=, date=0,drop=0,drop_ctw=);
/* make a ctw from a characteristic of the constituents of a ctw
For example, if &in_ctw has a fund's holdings
and &in_tv(&var) has the sector of the constituents,
the out_ctw gets the sector weights for each fund.
Alternatively, out_tv has one line per fund, with the sector weights as variables
parameters:
in_ctw(cticker,ticker,wgt) input constituents
in_tv(ticker,&var,...) constituent characteristics.  &var must be $8. or less
out_ctw(cticker,ticker,wgt) output, where each ticker is some value of &var
out_tv(ticker,val1,val2,...) partition form of output: tickers are the ctickers of &in
_ctw,
     val1... are the weights of corresponding to each value of &var
date - nonzero means in_ctw is (date,cticker,ticker,wgt)
drop - nonzero means drop and renormalize any constituents not assigned a &var in &tv
drop_ctw - optional dataset to put dropped constituents into
*/
   %if &out_ctw eq %then %let out_ctw = _c2v_ctw;


   proc sort data=&in_ctw out=&out_ctw;
```

```
   by ticker;

   data &out_ctw(drop=&var) %if &drop_ctw ne %then &drop_ctw(drop=&var);;
   length cticker $ 8 ticker $ 8;
   merge &out_ctw(in=ina) &in_tv(in=inb keep=ticker &var);
   by ticker;
   if ina;
   %if &drop_ctw ne %then %do;
      if not inb or &var eq '' then output &drop_ctw;
   %end;
   ticker = &var;
   %if &drop eq 0 %then %do;
      output &out_ctw;
   %end;
   %else %do;
      if ticker ne '' then output &out_ctw;
   %end;
   run;

   %ctw_dupo(in_ctw=&out_ctw,date=&date);
   %ctwnorm(in_ctw=&out_ctw,date=&date);

   %if &out_tv ne %then %do;
      proc transpose data=&out_ctw(rename=( ticker=t cticker=ticker)) out=&out_tv(keep
=ticker _numeric_);
      var wgt;
      id t;
      by ticker;
      run;
   %end;
   %if &out_ctw eq _c2v_ctw %then %trash(_c2v_ctw);;

%mend;

%macro ctwXtv(in_ctw=, in_tv=,var=, out_ctw=, map=);
/* ctw X tv - cross product of ctw and asset type
For example, if &in_ctw has a funds' holdings and
&in_tv has the sector of the constituents
then out_ctw gets a separate cticker for each fund X sector
Parameters:
in_ctw(cticker,ticker,wgt) input constituents
in_tv(ticker,&var,...) constituent characteristics.  &var must be $8. or less
var - name of variable containing the asset type designation
out_ctw(cticker,octicker,&var,ticker,wgt) output where cticker is a constructed value,
 octicker is the old cticker, etc.
map(cticker,octicker,&var) map of constructed ctickers to the cticker X &var
*/

   proc sort data=&in_ctw out=_ctvw;
   by ticker;

   data _ctvw;
   merge _ctvw(in=ina) &in_tv(keep=ticker &var);
   by ticker;
   if ina;
   if &var eq "" then &var = '_UNKNOWN';
   run;

   proc sort data=_ctvw;
   by cticker &var;
   run;

   data &out_ctw(keep=cticker %if &map eq %then octicker &var; ticker wgt) %if &map ne
%then &map(keep=cticker octicker &var);;
   length cticker $8;
   retain _count 0 cticker;
   set _ctvw(rename = (cticker=octicker));
   by octicker &var ;
   if first.&var then do;
      _count = _count + 1;
      cticker = compress("C_"||put(_count,6.));
      %if &map ne %then output &map;;
      end;
   output &out_ctw;
   run;

   %ctwnorm(in_ctw=&out_ctw);

%mend;


%macro mkctw(cticker=MKCTW, in_tv=, tickers=, tick_ma=, wgt=1,wgts=, wgt_ma=, out_ctw=
);
 /* make a ctw (with only one cticker)
  parameters
  cticker - value of cticker
  (choose tickers | tick_ma | in_tv)
  tickers - list of tickers
  tick_ma - macro array prefix for tickers
  in_tv(ticker) - dataset of tickers
  (choose wgt|wgts|wgt|ma)
  wgt - constant wgt
  wgts - list of weights
  wgt_ma - macro array prefix for weights
  out_ctw
  */
   %local i;
   %if &tick_ma eq %then %do;
      %global _aa0;
      %let _aa0 = 0;
      %let tick_ma = _aa;
   %end;

   %if &tickers ne %then %do;
      %l2ma(list=&tickers,ma=_aa);
   %end;
   %else %if &in_tv ne %then %do;
      %ds2ma(in_ds=&in_tv, var=ticker, ma=_aa);
   %end;

   %if "&wgts" ne "" %then %do;
      %l2ma(list=&wgts,ma=_ab);
      %let wgt_ma = _ab;
   %end;
   data &out_ctw;
   format cticker ticker $8. wgt 8.4;
   %if %eval(&&&tick_ma.0 > 0) %then %do;
      cticker = "&cticker";
      wgt = 1;
      %do i = 1 %to &&&tick_ma.0;
         ticker = "&&&tick_ma&i";
         %if &wgt_ma ne %then wgt = &&&wgt_ma&i;;
         output;
      %end;
   %end;
   %else %do;
      if 1=2;
   %end;
   run;
```

```
%mend;

%macro p2ctw(in_tv=, vars=, out_ctw=, wgt=1, norm=1, other=);
 /* turn partitioning to ctw
  parameters
  in_tv(ticker,var1,...)
  vars - list of variables that form partitioning
  out_ctw(cticker,ticker,wgt)
  wgt - variable in in_tv that determines weight of this ticker (defaults to equal wei
ghts)
  norm - normalize results (default is 1)
  other - name of cticker to assign remaining weights of assets to
  */

   data &out_ctw(keep=cticker ticker _wgt rename=(_wgt=wgt));
   set &in_tv;
   array _x &vars;
   format cticker $8.;
   if &wgt eq . then  _mwgt = 0; else  _mwgt = &wgt;
   %if &other ne %then %do;
      _sum=0;
      do over _x;
         if _x ne . then _sum = _sum+_x;
         end;
      if _sum < 1 then do;
         _wgt=_mwgt*( 1-_sum);
         cticker = "&other";
         if _wgt ne 0 then output;
         end;
   %end;
   do over _x;
      call vname(_x,cticker);
      _wgt = _mwgt * _x;
      if _wgt ne 0 and _wgt ne . then output;
      end;
   run;
   %if &norm ne 0 %then %ctwnorm(in_ctw=&out_ctw);

%mend;

%macro ctw2p(in_ctw=,out_tv=);
/* turn ctw to tv with tickers as partition variables
 in_ctw(cticker,ticker,wgt)
 out_tv(cticker as ticker,ticker1,ticker2,...)
 basically an l2wtran with cticker renamed to ticker
*/
   proc sort data=&in_ctw out=&out_tv;
   by cticker;
   run;

   proc transpose data=&out_tv out=&out_tv(keep=cticker _numeric_ rename=(cticker=tick
er));
   var wgt;
   id ticker;
   by cticker;
   run;

   %dsnonull(in_ds=&out_tv);

%mend;

%macro pXp2p(in_tv=, pa=, pb=, pc=, vara=, varb=, varc=, out_tv=, let_a=, let_b=, defa
ult=);
 /* cross product of two partitionings
  parameters
  in_tv(ticker,var1,var2,...)
  pa - macro array prefix for 1st partitioning
  pb - macro array prefix for 2nd partitioning
  pc - macro array prefix for output partitioning
  out_tv(ticker,var1,...)
  let_a letters to take from 1st partitioning for names of vars in pc
  let_b letters to take from 2nd partitioning for names of vars in pc
  default - partition name for unassigned
  alternatively, instead of pa,pb,pc, classification variables vara,varb,varc can be u
sed
  */

  %if &out_tv eq %then %let out_tv = &in_tv;

  %local i j k x y a z;
  %if &vara eq %then %do;
     %let k = 0;
     %global &pc.0;
     %global &pc._;
     %let &pc._ = ;
     %let &pc.0 = 0;
     %if &let_b ne %then %let a = %eval(8 - &let_b);
     %do i = 1 %to &&&pa.0;
        %do j = 1 %to &&&pb.0;
           %let k = %eval(&k + 1);
           %global &pc&k;
           %let x = &&&pa.&i;
           %let y = &&&pb.&j;
           %if &let_a ne %then %do;
              %if %eval(%length(&x) > &let_a) %then %let x = %substr(&x,1,&let_a);
           %end;
           %if &let_b ne %then %do;
              %if %eval(%length(&x) > &a) %then %let x = %substr(&x,1,&a);
              %if %eval(%length(&y) > &let_b) %then %let y = %substr(&y,1,&let_b);
           %end;
           %let z = %trim(&x)%trim(&y);
           %if %eval(%length(&z) > 8) %then %let z = %substr(&z,1,8);
           %let &pc.&k = &z;
           %if &let_a ne %then %do;
              %if %eval(%length(&x) > &let_a) %then %let x = %substr(&x,1,&let_a);
           %end;
           %if &let_b ne %then %do;
              %if %eval(%length(&x) > &a) %then %let x = %substr(&x,1,&a);
              %if %eval(%length(&y) > &let_b) %then %let y = %substr(&y,1,&let_b);
           %end;
           %let z = %trim(&x)%trim(&y);
           %if %eval(%length(&z) > 8) %then %let z = %substr(&z,1,8);
           %let &pc._ = &&&pc._ &z;
        %end;
     %end;
     %let &pc.0 = &k;

     data &out_tv(drop=_max);
     set &in_tv;
     format &pc $8.;
     %let k = 0;
     _max = 0;
     %do i = 1 %to &&&pa.0;
        %do j = 1 %to &&&pb.0;
           %let k = %eval(&k+1);
           if &&&pa&i eq . or &&&pa&i eq 0 or &&&pb&j eq . or &&&pb&j = 0
              then &&&pc&k = 0;
           else
              &&&pc&k = &&&pa&i * &&&pb&j;
           if &&&pc&k ne . and &&&pc&k > _max then do;
```

```
            &pc = "&&&pc&k";
            _max = &&&pc&k;
         end;
      %end;
   %end;
  %end;
  %else %do;
     data &out_tv(drop= _x _y);
     set &in_tv;
     format _x _y &varc $8.;
     _x = left(trim(&vara));
     _y = left(trim(&varb));
     %if &let_a ne %then %do;
        if length(_x) > &let_a then _x = substr(_x,1,&let_a);
     %end;
     %if &let_b ne %then %do;
        if length(_x) > &a then _x = substr(_x,1,&a);
        if length(_y) > &let_b then _y = substr(_y,1,&let_b);
     %end;
     &varc = left(trim(_x)) || left(trim(_y));
     &varc=left(&varc);
     if length(&varc) > 8 then &varc = substr(&varc,1,8);
     %let pc = &varc;
  %end;
  %if &default ne %then %do;
     if &pc = "" then do;
        &pc = "&default";
        &default = 1;
     end;
  %end;
  run;

%mend;



%macro tvfilter(in_tv=,  tickers=,  in_t=,  vars= ,vards=, out_tv=);
 /* tvfilter - removes tickers and/or variables from ticker-keyed dataset
  parameters
  in_tv(ticker,var1,...varn) source
  choose tickers | in_t
  tickers - list of tickers to keep
  in_t(ticker) dataset of tickers to keep
  choose vars | vards
  vars - list of variables to keep (ticker is implied)
  vards(var) - ds of variables to keep (ticker is implied)
  var and vards are exclusive, if neither present all attributes are kept
  out_tv(ticker,var1,...varn)
  */
   %if &out_tv eq %then %let out_tv = &in_tv;
   %if &tickers ne  %then %do;
      %l2ds(list=&tickers,var=ticker,out_ds=_tv1);
      %let in_t=_tv1;
   %end;
   %if &vars ne  %then %do;
      %l2ma(list=&vars,ma=_cv);
   %end;
   %else %if &vards ne %then %do;
      %ds2ma(in_ds=&vards,var=var,ma=_cv);
   %end;
   %else %do;
      %dsv2ma(in_ds=&in_tv,drop=ticker,ma=_cv);
   %end;
   proc sql;
   create table &out_tv as
      select ticker, %ma21(ma=_cv,sep=%quote(,))
      from &in_tv
      %if &in_t ne %then %do;
         where ticker in (select ticker from &in_t)
            %end;;
   %if &tickers ne %then %trash(_tv1);;

%mend;



%macro tv2ctw(in_tv=, vars=, out_ctw=);
 /* converts ticker-vars to cticker-ticker-wgt
  parameters
  in_tv(ticker,...&var...)
  vars - list of variables corresponding to ctickers
  out_ctw(cticker,ticker,wgt)
  */

   %if &vars ne %then %do;
      %kvv2kkv(in_ds=&in_tv,out_ds=&out_ctw,key=ticker,
               vars=&vars,out_key=cticker,out_var=wgt);
   %end;
   %else %do;
      data &out_ctw;
      format cticker ticker $8.;
      set &in_tv(keep=ticker);
      wgt=1;
      cticker = "TV2CTW";
      run;
   %end;

%mend;



%macro v2ctw(in_tv=, in_dtt=, var=, wgt=, out_ctw=, out_dtt=);
 /* variable contents to portfolio collection
  each value of &var becomes a cticker in a ctw,
  weighted by &wgt (defaults to equi-weighted)
  parameters
  in_tv(ticker,...&var...&wgt)
  in_dtt(date, ticker1,...)
  var - name of string variable in &in_tv
  wgt - name of numeric variable in &in_tv holding weights
  out_ctw(cticker,ticker,wgt)
  out_dtt(date,ticker1,...)
  */
   data &out_ctw(keep= &var ticker wgt rename=(&var=cticker));
   set &in_tv(keep=ticker &var &wgt);
   %if &wgt ne %then wgt=&wgt; %else wgt=1;;
   run;

   %ctw_dupo(in_ctw=&out_ctw);
   %ctwnorm(in_ctw=&out_ctw);
   %if &out_dtt ne %then %ctwret(in_ctw=&out_ctw,in_dtt=&in_dtt,out_dtt=&out_dtt);;

%mend;

%macro ctw_topn(in_ctw=, n=, out_ctw=);
/* top weighted assets in portfolio
in_ctw(cticker,ticker,wgt) - portfolios
n - get top &n assets in each portfolio
```

```
out_ctw(cticker,ticker,wgt) - resultant partial portfolios
*/
   proc sort data=&in_ctw out=&out_ctw;
   by cticker descending wgt;
   run;

   data &out_ctw(drop=n);
   retain n 0;
   set &out_ctw;
   by cticker;
   if first.cticker then n = 0;
   n = n + 1;
   if n <= &n then output;
   run;

%mend;

%macro ctwcombo(in_ctw=, in_tv=, var=, vars=, out_ctw=, out_def=, wgt=1);
 /* create combined portfolios
  creates new porfolios q = p X v
  for each p in in_ctw and v in var
  the in_tv must have an assignment of 1 or 0
  for each ticker / v to indicate inclusion of this ticker
  in v portfolios.
  parameters
  in_ctw(cticker,ticker,wgt)
  in_tv(ticker,...var1,var2,...,varn,...)
  choose var|vars
  var - variable whose values are used to form partition
  vars - list of variables in in_tv that have 1's if ticker is in collection, 0 otherw
ise
  out_ctw(cticker,ticker,wgt) ctickers are artificially constructed, and are the _pfno
's in out_def
  out_def(cticker,var,_pfno) the mapping rule back from out_ctw
  */

   %if &var ne %then %do;
      proc sql noprint;
      create table _a9 as
         select &var as cticker, ticker, &wgt as wgt
         from &in_tv
         order by cticker;
      %ctwnorm(in_ctw=_a9);
   %end;
   %else %do;
      %p2ctw(in_tv=&in_tv,vars=&vars,out_ctw=_a9,wgt=&wgt);
   %end;

   %ctw_x(in_ctw1=&in_ctw,in_ctw2=_a9,out_ctw=&out_ctw,out_def=&out_def);
   %trash(_a9);
%mend;

%macro twsum2mv(in_ds=,mv=);
/* sum-of-ticker-times-weight expression to a macro variable
in_ds(ticker,wgt)
mv - name of macro variable
*/
    %global &mv;
    data _null_;
    length _x $10 _y $20;
    set &in_ds end=eof;
    _x = compress("_MVL"||put(_n_,6.));
    _y = trim(compress(ticker||'*'||put(wgt,best.)));
    call symput(_x,trim(_y));
    if eof then call symput("_MVL0",compress(put(_n_,6.)));
    run;
    %let &mv = %ma21(ma=_mv1,sep=%quote(+));

%mend;


%macro ctw_rect(in_ctw=,out_ctw=);
/* make ctw a rectangle
ensures that all tickers in one cticker are in all ctickers
*/

   %if &out_ctw eq %then %let out_ctw = &in_ctw;

   proc sort data=&in_ctw out=&out_ctw;
   by cticker ticker;
   run;

   proc transpose data=&out_ctw out=&out_ctw;
   var wgt;
   id ticker;
   by cticker;
   run;

   %dsnonull(in_ds=&out_ctw);

   proc transpose data=&out_ctw out=&out_ctw(rename=(col1=wgt)) name=ticker;
   by cticker;
   run;

%mend;

%macro ctw_turnover(ctw1=,ctw2=,out_ds=);
/* portfolio turnover
input:
ctw1(cticker,ticker,wgt) - initial portfolios
ctw2(cticker,ticker,wgt) - final portfolios
out_ds(cticker, turnover) - turnover of each portfolio
*/
   proc sql noprint;
   create table c1 as
      select distinct cticker
      from &ctw1;
   create table c2 as
      select distinct cticker
      from &ctw2;
   create table ct1 as
      select a.*
      from c2, &ctw1 a
      where a.cticker=c2.cticker
      order by a.cticker,a.ticker;
   create table ct2 as
      select a.*
      from c1, &ctw2 a
      where a.cticker=c1.cticker
      order by a.cticker,a.ticker;

   data &out_ds(keep=cticker turnover);
   merge ct1(in=in1 rename=(wgt=wgt1)) ct2(in=in2 rename=(wgt=wgt2));
   by cticker ticker;
   retain turnover;
   if wgt1 = . then wgt1 = 0;
   if wgt2 = . then wgt2 = 0;
   if first.cticker then turnover = 0;
   if wgt1 > wgt2 then turnover = turnover + wgt1 - wgt2;
```

```
    if last.cticker then output;
    run;

%mend;
```

date.macro


```
/* Date operations
include $FPPCODE/utilities/date.macro
 Generally dates are assumed to be numeric variables yyyymmdd = 10,000*year+100*month+
day or yyyymm
 These can be converted to SAS dates for graphing, etc. using the conversion functions
 included.
 N.B. The year and month functions are not smart about number of days in month.
*/

options yearcutoff=1950;

%macro n2sasdt(in_date=date, out_date=date, in_ds=, out_ds=);
/* numeric to sasdate
 converts numeric yyyymmdd to sas date variable
 if in_ds is present converts whole dataset.
 Otherwise, can be used inside a data step

Example 1:
 data x;
 input date val;
 cards;
20010131 1
20010201 2
;
%n2sasdt(in_ds=x);
proc print data=x;run;
OBS          DATE    VAL
 1     2001-01-31     1
 2     2001-02-01     2

 Example 2:

 data x;
 input date val;
 cards;
20010131 1
20010201 2
;
 data y;
 format sasdate yymmdd10.;
 set x;
 sasdate=%n2sasdt;
 run;
 proc print data=y;run;
 OBS       SASDATE      DATE      VAL
 1     2001-01-31    20010131     1
 2     2001-02-01    20010201     2
*/
   %if &in_ds ne %then %do;
      %if &out_ds eq %then %let out_ds = &in_ds;
      data &out_ds;
      set &in_ds;
      format &out_date yymmdd10.;
      &out_date= input(put(&in_date,8.),yymmdd10.);
      run;
   %end;
   %else %do;
      input(put(&in_date,8.),yymmdd10.);
   %end;
%mend;

%macro sasdt2n(in_date=date, out_date=date, in_ds=, out_ds=);
/* sasdate to numeric
 converts sas date variable to numeric yyyymmdd
 if in_ds is present converts whole dataset.
 Otherwise, can be used inside a data step.

 Example 1:
 data x;
 input date yymmdd10. val;
 cards;
2001-01-31 1
2001-02-01 2
;
%sasdt2n(in_ds=x);
 proc print data=x; run;
 OBS         DATE    VAL
 1      20010131     1
 2      20010201     2

 Example 2:
 data x;
 input date yymmdd10. val;
 cards;
2001-01-31 1
2001-02-01 2
;
data y;
format date 8.;
set x;
date=%sasdt2n;
proc print data=y; run;
 1      20010131     1
 2      20010201     2
*/

   %if &in_ds ne %then %do;
      %if &out_ds eq %then %let out_ds = &in_ds;
      data &out_ds;
      set &in_ds;
      format &out_date 8.;
      &out_date = input(put(&in_date,mmddyy4.),4.)+10000*input(put(&in_date,year4.),4.
);
      run;
   %end;
   %else %do;
      input(put(&in_date,mmddyy4.),4.)+10000*input(put(&in_date,year4.),4.);
   %end;
%mend;

%macro dt_plusd(in_date=, days=, out_date=);
/* add days to numeric date - macro variable version
  in_date - numeric date
  days number of days to add
  out_date name of macro variable to output date to

Example:
%dt_plusd(in_date=20010101,days=44,out_date=x);
%put &x;
20010214
*/
   %global  &out_date;
   data _null_;
   format y 8.;
   numdt=&in_date;
   sasdt=input(put(numdt,8.),yymmdd10.);
   sasdt=sasdt+&days;
   md = input(put(sasdt,MMDDYY4.),4.);
   y = input(put(sasdt,year4.),4.);
   call symput("&out_date",put(10000*y+md,8.));
```

```
   run;
%mend;

%macro df_plusd(in_date=date, days=);
/* add days to numeric date - function version
  in_date - variable containing numeric date
 days - days to add
 data x;
 input date val;
 cards;
20010131 1
20010201 2
;
 data y;
 set x;
 date_p3 = %df_plusd(days=3);
 run;
 proc print data=y;run;
 OBS      DATE      VAL    DATE_P3
 1     20010131     1     20010203
 2     20010201     2     20010204
*/
   put(10000*input(put(input(put(&in_date,8.),yymmdd10.)+&days,year4.),4.)+
       input(put(input(put(&in_date,8.),yymmdd10.)+&days,mmddyy4.),4.),8.)
%mend;


%macro dt_plusm(in_date=, months=, out_date=, dd=1);
/* add months to numeric date - macro variable version
  in_date - numeric date
  months number of months to add
  out_date name of macro variable to output date to
 dd 0 means date is really yyyymm 1 means yyyymmdd

Example:
%dt_plusm(in_date=200101,months=-3,out_date=back3);
%put &back3;
200010
*/
   %global &out_date;
   %local y m d;
   %let y = %substr(&in_date,1,4);
   %let m = %substr(&in_date,5,2);
   %if &dd ne 0 %then %let d = %substr(&in_date,7,2); %else %let d = ;
   %let m = %eval(&m + &months);
   %do %while (%eval(&m > 12));
      %let y = %eval(&y + 1);
      %let m = %eval(&m - 12);
   %end;
   %do %while (%eval(&m < 1));
      %let y = %eval(&y - 1);
      %let m = %eval(&m + 12);
   %end;
   %if %length(&m) eq 1 %then %let m = 0&m;
   %let &out_date = &y&m&d;
%mend;

%macro df_plusm(in_date=date, months=);
/* add months to numeric date - function version
  in_date - numeric date
  months number of months to add
 call this inside a data step:
 data x;
 set y;
 date_p3 = %df_plusm(months=3);
 this doesn't work at end of month.  To get around this, do this
 datep=%df_plusd(days=1);
 datep=%df_plusm(in_date=datep,months=n);
 datep=%df_plusd(in_date=datep,days=-1);
*/
   (int((((int((&in_date)/10000)*12) + (int((&in_date)/100) - int((&in_date)/10000) *1
00) + &months) - 1)/12))*10000 +
      (((int((&in_date)/10000)*12) + (int((&in_date)/100) - int((&in_date)/10000) *100
) + &months) - 12*(int((((int((&in_date)/10000)*12) + (int((&in_date)/100) - int((&in_
date)/10000) *100) + &months) - 1)/12))) * 100 +
      ((&in_date)- int((&in_date)/100)*100)
%mend;


%macro dt_plusy(in_date=, years=, out_date=);
/* add years to numeric date - macro variable version
  in_date - numeric date
  years -  number of years to add
  out_date name of macro variable to output date to

Example:
%dt_plusy(in_date=200101,years=-3,out_date=back3);
%put &back3;
199801
*/
   %local y md;
   %let y = %substr(&in_date,1,4);
   %let md = %substr(&in_date,5,4);
   %let y = %eval(&y + &years);
   %let &out_date = &y&md;

%mend;

%macro df_plusy(in_date=date, years=);
/* add years to numeric date - function version
  in_date - numeric date
  years number of years to add
 call this inside a data step:
 data x;
 set y;
 date_p3 = %df_plusm(years=3);
*/
   (int(&in_date/10000) + &years)*10000 +
      &in_date - int(&in_date/10000)*10000
%mend;


%macro dt_today(out_date);
/* today as numeric date - macro variable version
out_date - name of macro variable to put current yyyymmdd date into
Example:
%dt_today(kyo);
%put &kyo;
20010220
*/
   %global &out_date;
   data _null_;
   date=today();
   date=%sasdt2n;
   call symput("&out_date",trim(left(put(date,8.))));
   run;
```

```
%mend;


%macro df_eom(in_date=date);
/* function to calculate last day of month
Example:
 data x;
 input date val;
 cards;
20010131 1
20010201 2
;
 data y;
 set x;
 eom=%df_eom;
 run;
 proc print data=y;run;
OBS      DATE      VAL      EOM
 1     20010131     1     20010131
 2     20010201     2     20010228
*/

   put(10000*input(put(input(put(((int(((int((&in_date)/10000)*12) + (int((&in_date)/1
00) - int((&in_date)/10000) *100) )/12))*10000 +
 (((int((&in_date)/10000)*12) + (int((&in_date)/100) - int((&in_date)/10000) *100) + 1
)
 - 12*(int(((int((&in_date)/10000)*12) + (int((&in_date)/100)
 - int((&in_date)/10000) *100) )/12))) * 100 + 1),8.),yymmdd10.)-1,year4.),4.)+
       input(put(input(put(((int(((int((&in_date)/10000)*12) + (int((&in_date)/100) -
int((&in_date)/10000) *100) )/12))*10000 +
 (((int((&in_date)/10000)*12) + (int((&in_date)/100) - int((&in_date)/10000) *100) + 1
)
 - 12*(int(((int((&in_date)/10000)*12) + (int((&in_date)/100)
 - int((&in_date)/10000) *100) )/12))) * 100 + 1),8.),yymmdd10.)-1,mmddyy4.),4.),8.)


%mend;

%macro day_diff(date1=, date2=, mv=);
/* difference of two days;
macro variable &mv gets date1-date2
Example:
%day_diff(date1=20010131,date2=20001130,mv=twomonth);
%put &twomonth;
62
*/
   %global &mv;
   data _null_;
   date1=&date1;
   date1=%n2sasdt(in_date=date1);
   date2=&date2;
   date2=%n2sasdt(in_date=date2);
   call symput("&MV",left(trim(put(date1-date2,8.))));
   run;
%mend;

%macro ym2ma(from_dt=, to_dt=, ma= );
/* yyyymm date range to macro array.
from - from date
to - to date
ma - prefix of macro array

Example:
%ym2ma(from_dt=200011, to_dt=200102, ma=m);
%put &m0 &m1 &m2 &m3 &m4;
4 200011 200012 200101 200102
*/

   %local _y _m _m2 _fy _ty _i;
   %let _fy = %eval(&from_dt / 100 );
   %let _ty = %eval(&to_dt / 100);
   %global &ma.0;
   %let &ma.0 = 0;
   %do _y = &_fy %to &_ty;
      %do _m = 1 %to 12;
         %if %eval(&_m < 10) %then %let _m2 = 0&_m;
         %else %let _m2 = &_m;
         %if %eval(&_y.&_m2 >= &from_dt)  and %eval(&_y.&_m2 <= &to_dt) %then %do;
            %let &ma.0 = %eval(&&&ma.0 + 1);
            %global &ma.&&&ma.0;
            %let &ma.&&&ma.0 = &_y.&_m2;
         %end;
      %end;
   %end;
%mend;


%macro fwd_date(in_ds=, out_ds=, end_date=,date_ds=);
/* slides all dates forward one
Each date gets moved forward to the next date in &in_ds (or if &date_ds exists, the ne
xt date in &date_ds)
if end_date is present, that's what the records in in_ds with the last date
get mapped to.  Otherwise they get removed

Example 1:
 data x;
 input date val;
 cards;
20010131 1
20010201 2
;
%fwd_date(in_ds=x,out_ds=y);
proc print data=y;run;
 OBS      DATE      VAL
 1     20010201     1

Example 2:
%fwd_date(in_ds=x,out_ds=y,end_date=20010203);
proc print data=y;run;
 OBS      DATE      VAL
 1     20010201     1
 2     20010203     2

*/

   %if &date_ds eq %then %let date_ds=&in_ds;

   proc sql;
   create table _dates as
      select distinct date from &date_ds
      order by date;

   %local max_date;
   %if &end_date ne %then %do;
      proc sql noprint;
      select max(date) into :max_date
         from _dates;
   %end;
```

```
   data _dates;
   merge _dates(in=in1) _dates(firstobs=2 rename=(date=_nextdt) in=in2);
   if in2;
   run;

   %if &end_date ne %then %do;
      proc sql;
      insert into _dates(date,_nextdt) values (&max_date,&end_date);
   %end;

   data &out_ds(drop=date _olddate rename=(_nextdt=date));
   merge  _dates(in=_ind) &in_ds(in=_inds);
   by date;
   retain _olddate;
   if not _ind then _nextdt=_olddate;
   else _olddate = _nextdt;
   if _inds and _ind then output;
   run;

%mend;

%macro ym2eom(in_ds=,out_ds=);
/* convert yyyymm date to yyyymmdd where dd is last day of month

Example:
 data x;
 input date val;
 cards;
200101 1
200102 2
;
run;
%ym2eom(in_ds=x);
 proc print data=x;run;
OBS         DATE    VAL
 1      20010131     1
 2      20010228     2
*/

   %if &out_ds eq %then %let out_ds=&in_ds;
    data &out_ds;
    format date 8.;
    set &in_ds;
    date = %df_eom(in_date=100*date+1);
    run;

 %mend;

 %macro date2sasdconst(in_date=,out_date=);
 /* convert macro variable yyyymmdd to sas date macro variable
 */
    %global &out_date;
    data _null_;
    date=&in_date;
    date=%n2sasdt;
    call symput("&out_date",left(trim(put(date,date.))));
    run;

 %mend;
```

```
   x 'setenv SYB_SAS_V6PASSTHRU 1';

 /* sybase - sas utilities
*/

 /* internal use */ %macro _dbparms(db=);

   %global database dbserver dbuser  dbpass;

   %let db = %upcase(&db);

   %if &db eq MIAS %then %do;
      %let database=mias_R2;
      %let dbserver = SYBTEXAS;
      %let dbuser = risk;
      %let dbpass = risk11;
   %end;
   %else %if &db eq CRISP %then %do;
      %let database=crisp;
      %let dbserver = SYBTEXAS;
      %let dbuser = crispadm;
      %let dbpass = bobvick;
   %end;
   %else %if &db eq POD %then %do;
      %let database=pod;
      %let dbserver = SYBCAUCHY;
      %let dbuser = wateradm;
      %let dbpass = wateradm;
   %end;


%mend;


%macro ds2table(db=,in_ds=,table=);
/* move data in a sas dataset to a sybase table
in_ds - dataset
table - name of table
inserts all rows and columns of dataset into table
table must not already exist
*/

/* build table definition */

   %_dbparms(db=&db);

   proc contents data=&in_ds out=_dbc noprint;
   run;

proc sort data=_dbc;
   by varnum;
   run;

%local i;

data _null_;
   set _dbc end=eof;
   mname = "fld"||left(trim(put(_n_,8.)));
   call symput(mname,name);
   put "put " mname " -> " name;
   if eof then call symput("FLD0",trim(left(put(_n_,8.))));
   run;
%put &fld0;

proc dbload;
      dbms=&dbms;
      data=&in_ds ;
   server="&dbserver";
   database="&database";
   table="&table";
   user="&dbuser";
   password="&dbpass";
   bulkcopy=Y;
   commit=0;
   limit=0;
   %do i = 1 %to &fld0; nulls &&fld&i=Y; %end;
   load;
   run;


%mend;


%macro dbgetpfx(db=,ds=);
/* prefix to getting data from sybase into a dataset
db - MIAS | CRISP | POD
ds - name of dataset to create
Example:
to get a fund name from MIAS,

%dbgetpfx(db=MIAS,ds=f)
 select name
 from funds
 where fund_id = '00256'
 );

NOTE: Be sure to leave out semicolon after macro call and put in paren-semicolon after
 SQL!

*/
   %_dbparms(db=&db);

   proc sql noerrorstop;
   connect to sybase as
      &database(
                server=&dbserver
                database=&database
                user=&dbuser
                pass=&dbpass
                );
   create table &ds as select * from connection
      to &database(

%mend;

%macro dbexec(db=, sql=);
 /*  execute  an sql statement
  db - MIAS | CRISP | POD
  */
   %_dbparms(db=&db);


  %local _excount;
  %let _execount = 5;

  /* make 5 attempts with a 5 second wait each if there is a deadlock */
  %do %until (&_excount eq 0);

     proc sql noerrorstop;
```

```
      connect to sybase as
         &database(
                   server=&dbserver
                   database=&database
                   user=&dbuser
                   pass=&dbpass
                   );
      execute (&sql)
      by &database;

      %if %eval(&sqlxrc eq -1205) %then %do;
         x 'sleep 5';
         %let _excount = %eval(&_excount - 1);
      %end;
      %else %do;
         %let _excount = 0;
      %end;
   %end;
   %if %eval(&sqlxrc ne 0) %then %put ERROR sqlxrc = &sqlxrc. &sqlxmsg;

%mend;

%macro dblibname(db=);
/* set up a database as a libname
db - one of MIAS, CRISP, POD
*/

   %_dbparms(db=&db);

   libname &db sybase user=&dbuser password = &dbpass database=&database server=&dbser
ver;


%mend;
```

```
 /* dataset utilities
%include $FPPCODE/utilities/macros.macro
*/

%macro l2wtran(in_ds=, out_ds=, id=ticker, by=, var=, sort=0);
/* long to wide transposer
in_ds(&id,&var,&by) input dataset
out_ds(&by,id1,id2,...) output dataset
id - name of variable whose values become column names
by - optional variable(s) that will form key to tranposed dataset
var - name of variable containing values to put into &id columns
sort - nonzero indicates that input dataset must be sorted by &by first.

Example:
 data x;
 format cticker  ticker $8. wgt 6.4;
 input cticker ticker wgt;
 cards;
FMAGX T .5
FMAGX XOM .5
FCNTX T .3
FCNTX R .7
;
%l2wtran(in_ds=x, out_ds=y, id=ticker,by=cticker,var=wgt,sort=1);
 proc print data=y; run;
OBS     CTICKER          T          R        XOM
 1      FCNTX        0.3000     0.7000       .
 2      FMAGX        0.5000       .          0.5000
*/

   %if &sort ne 0 %then %do;
      proc sort data=&in_ds;
      by &by;
      run;
   %end;
   %if &out_ds eq %then %let out_ds = &in_ds;
   proc transpose data=&in_ds(keep=&var &id &by) out=&out_ds(keep=&by _numeric_);
   var &var;
   id &id;
   %if &by ne %then by &by;;
   run;
%mend;

%macro lw2wwtrn(in_ds=, out_ds=, id=, by=, sort=0, id_map=, maa=, mab=, prefixes=A B);
/* transform a long and wide dataset to a very wide one
in_ds - dataset to transform
out_ds - resultant dataset defaults to in_ds
id - id variable for long part
by - by variable
(all other variables are considered the wide ones)
sort - nonzero if in_ds should be sorted first
id_map (oname,oval,nval) - dataset resultant variable names:
 oname is either the id variable name or _ID for the wide variables
 oval is the value for the "id" variable or the name of one of the wide variables
 nval are the mapped values, which are concatenated to form the resultant variable nam
es
maa - output  macro array of  &id variables
mab - output macro array of wide variables
prefixes -  a list of 2 items: the prefix for the nval for the id variable and
 a prefix for the nval for the wide variables
For example:
 date fticker a b c
990101 F1     11 12 13
990102 F1     21 22 23
990101 F2     31 32 33

gets transformed by %lw2wwtrn(in_ds=x, id=fticker,by=date,sort=1,id_map=m,prefixes=x y
) to:
date   X1Y1 X1Y2 X1Y3 X2Y1 X2Y2 X2Y3
990101 11    12   13   31   32   33
990102 21    22   23    .    .    .

with m=:
oname oval nval
FTICKER F1 X1
FTICKER F2 X2
_ID A Y1
_ID B Y2
_ID C Y3

*/
   %let id = %upcase(&id);
   %if &out_ds eq %then %let out_ds = &in_ds;
   %w2ltran(in_ds=&in_ds,out_ds=_dt3,by=&by &id,var=_var,id=_id,sort=&sort);
   %if &id_map eq %then %let id_map = _id_map;

   %ll2wwtrn(in_ds=_dt3,out_ds=&out_ds,by=&by,ids=&id _ID,var=_var,id_map=&id_map,pref
ixes=&prefixes);
   %if &maa ne %then %do;
      data _ds1;
      set &id_map(where=(oname="&id"));
      run;
      %ds2ma(in_ds=_ds1,var=oval,ma=&maa);
      %trash(_ds1);

   %end;
   %if &mab ne %then %do;
      data _ds1;
      set &id_map(where=(oname="_ID"));
      run;
      %ds2ma(in_ds=_ds1,var=oval,ma=&mab);
      %trash(_ds1);
  %end;

%mend;


%macro ll2wwtrn(in_ds=, out_ds=, var=, by=, ids=, sort=0, id_map=, prefixes=);
/* extension of l2wtran with more than one id
in_ds - dataset to transform
out_ds - resultant dataset defaults to in_ds
ids - list of id variables
by - by variable
sort - nonzero if in_ds should be sorted first
id_map (oname,oval,nval) - dataset resultant variable names:
 oname is the id variable name
 oval is the value for the "id" variable
 nval are the mapped values, which are concatenated to form the resultant variable nam
es
prefixes -  the prefixes for the mapped variable names
&prefixes must have the same number of elements as &ids
For example:
 date fticker sticker wgt
 990101 F1     S1      11
 990102 F1     S2      12
 990101 F2     S1      21

gets transformed by %ll2wwtrn(in_ds=x, ids=fticker sticker,by=date,var=wgt, sort=1,id_
map=m,prefixes=x y) to:
date   X1Y1 X1Y2 X2Y1
```

```
990101 11     .    21
990102 .      12    .

with m=:
oname oval nval
FTICKER F1 X1
FTICKER F2 X2
STICKER S1 Y1
STICKER S2 Y2

*/
   %if &sort ne 0 %then %do;
      proc sort data=&in_ds;
      by &by;
      run;
   %end;
   %if &out_ds eq %then %let out_ds = &in_ds;
   %let ids = %upcase(&ids);
   %let prefixes = %upcase(&prefixes);

   %l2ma(list=&ids,ma=_id);
   %l2ma(list=&prefixes,ma=_pr);
   %local i;

   %if &id_map eq %then %let id_map = _id_map;
   proc sql noprint;
   create table &id_map(oname char(8), oval char(8),nval char(8));

   data _dt0;
   set &in_ds;
   run;

   %do i =1 %to &_pr0;
      %varenum(in_ds=_dt0,out_ds=_dt1,var=&&_id&i,prefix=&&_pr&i,id=nval);
      %v_replac(in_ds=_dt0,in_ref=_dt1,var=&&_id&i,repvar=nval);
      data &id_map;
      set &id_map _dt1(rename=(&&_id&i=oval) in =ina);;
      if ina then oname="&&_id&i";
      run;
   %end;

   data _dt2(drop=&ids);
   set _dt0;
   format _id1 $8.;
   _id1 = trim(left(&_id1))
      %do i=2 %to &_id0;
         || trim(left(&&_id&i))
      %end;;
   run;

   %l2wtran(in_ds=_dt2,out_ds=&out_ds,var=&var,by=&by,id=_id1,sort=1);

%mend;


%macro w2ltran(in_ds=, out_ds=, id=ticker, var=, by=, sort=0, nonull=1);
/* wide to long transposer
in_ds(&by,id1,id2,id3,...) input dataset
out_ds(&by,&id,&var) output dataset
id - name of variable that gets transposed column names
var - name of variable that get transposed column values
by - key of &in_ds
sort - indicates &in_ds must be sorted first.
nonull - nonzero indicates null values should be removed from out_ds

Example:
 data y;
 format CTICKER  $8. T R XOM 6.4;
 input CTICKER  $ T R XOM;
 cards;
FCNTX       0.3000    0.7000      .
FMAGX       0.5000       .      .     0.5000
;
%w2ltran(in_ds=y, out_ds=x, id=ticker,var=wgt,by=cticker,sort=1);
proc print data=x;run;
OBS     CTICKER      TICKER        WGT
 1      FCNTX         T          0.3000
 2      FCNTX         R          0.7000
 3      FMAGX         T          0.5000
 4      FMAGX         XOM        0.5000
*/

   %if &sort ne 0 %then %do;
      proc sort data=&in_ds;
      by &by;
      run;
   %end;
   %if &out_ds eq %then %let out_ds = &in_ds;
   proc transpose data=&in_ds out=&out_ds(keep = &by &id col1 rename=(col1=&var)) name
=&id;
   %if &by ne %then by &by;;
   run;
   %if &nonull ne 0 %then %do;
      data &out_ds;
      length &id $8;
      set &out_ds(where=(&var ne .));
      run;
   %end;
%mend;

%macro reformat(in_ds, vars, fmt);
/* reformats variables in dataset
in_ds(&vars,..) input dataset
vars-list of variables to format
fmt - format to be applied to all &var variables

Example:
proc print data=x;id cticker; run;
 CTICKER      TICKER        WGT
 FCNTX         T          0.3000
 FCNTX         R          0.7000
 FMAGX         T          0.5000
 FMAGX         XOM        0.5000
%reformat(x, wgt, 4.2);
proc print data=x;id cticker;run;
 CTICKER      TICKER       WGT
 FCNTX         T          0.30
 FCNTX         R          0.70
 FMAGX         T          0.50
 FMAGX         XOM        0.50
*/

   data &in_ds;
   set &in_ds;
   format &vars &fmt;
   run;

%mend;
```

```
%macro printds(ds, id, by);
/* simple dataset print
ds - name of dataset to print
id - use this column as output row identifier
by - passed through as by statement to proc print
prints to &ds.txt in current directory
*/

proc printto;run;
   filename outprt "&ds..txt";

proc printto new print=outprt;

proc print data=&ds;
   title "&ds";
   %if &id ne %then id &id;;
   %if &by ne %then by &by;;
   run;
proc printto;
   run;
%mend;


%macro trash(ds,lib);
 /* deletes temporary datasets
  ds - a list of datasets in the same library, without the lib name in front
  lib - name if library. Defaults to work

  Examples:
  %trash(a b c, l);
  deletes datasets l.a l.b and l.c

  %trash(x y z);
  deletes temporary datasets x y and z
*/

   %if &lib eq %then %let lib=work;

   proc datasets nolist lib=&lib;
   delete &ds;
   run;

%mend;

%macro varenum(in_ds=, out_ds=, var=ticker, prefix=, id=tickid);
 /* produce enumeration of distinct values of a variable
  this is useful if you're merging 2 datsets with the same columns...
  parameters
  in_ds(&var,...)
  out_ds(&var,&id) where &id is a 1-1 map to values of &var
  var - key variable in in_ds
  prefix - prefix of id variable.if blank, a numeric id variable is created

Example:
 data x;
 format cticker  ticker $8. wgt 6.4;
 input cticker ticker wgt;
 cards;
FMAGX T .5
FMAGX XOM .5
FCNTX T .3
FCNTX R .7
;
%varenum(in_ds=x,out_ds=y, prefix=X);
proc print data=y;id ticker;run;
```

```
 TICKER      TICKID
 R           X1
 T           X2
 XOM         X3
*/

   proc sql;
   create table &out_ds as
      select distinct &var as &var
      from &in_ds
      order by &var;

   data &out_ds;
   set &out_ds;
   %if &prefix ne %then %do;
      format &id $8.;
      &id = "&prefix" || left(trim(put(_n_,8.)));
   %end;
   %else %do;
      &id = _n_;
   %end;
   run;
%mend;

%macro dsrmnullvar(in_ds=,out_ds=,keep=);
/* remove variables from a dataset that have all rows null
*/
   %if &out_ds eq %then %let out_ds = &in_ds;
   %local i;

   %dsv2ma(in_ds=&in_ds, drop=&keep, ma=_dsv);

   %ma_live(in_ds=&in_ds, in_ma=_dsv, out_ma=,nonzero=0);

   data &out_ds;
   set &in_ds(keep= &keep %do i = 1 %to &_dsv0; &&_dsv&i %end;);
   run;

%mend;

%macro dsnonull(in_ds=, out_ds=, vars=,zeronull=1,anynull=1);
 /* removes null rows from a dataset
  parameters
  in_ds() input dataset
  out_ds() defaults to in_ds
  vars - list of variables in in_ds to remove nulls for. if omitted, all numeric varia
bles are de-nulled
  zeronull =1 means set all null values to zero.  Otherwise,
  anynull  = 1 means remove rows with  any nulls, otherwise remove rows with all  vari
ables null
Example:
 data y;
 format CTICKER  $8. T R XOM 6.4;
 input CTICKER  $ T R XOM;
 cards;
FCNTX       0.3000     0.7000      .
FMAGX       0.5000      .          0.5000
FMILX       0.1000     0.7000     0.2000
;
%dsnonull(in_ds=y,out_ds=x);
proc print data=x;id cticker;run;
 CTICKER          T          R          XOM
 FCNTX       0.3000     0.7000     0.0000
 FMAGX       0.5000     0.0000     0.5000
 FMILX       0.1000     0.7000     0.2000
```

```
%dsnonull(in_ds=y,out_ds=z,zeronull=0);
proc print data=z;id cticker;run;
 CTICKER          T          R          XOM
 FMILX       0.1000     0.7000     0.2000
*/
   %if &out_ds eq %then %let out_ds = &in_ds;

   %if &anynull eq 1 %then %do;
      data &out_ds;
      set &in_ds;
      array _x %if &vars eq %then _numeric_; %else &vars;;
      do over _x;
         %if &zeronull eq 1 %then %do;
            if _x = .  then _x = 0;
         %end;
         %else %do;
            if _x ne .;
         %end;
         end;
      run;
   %end;
   %else %do;
      data &out_ds(drop=_ok);
      set &in_ds;
      array _x %if &vars eq %then _numeric_; %else &vars;;
      _ok = 0;
      do over _x;
         if _x ne . then do;
            _ok = 1;
            end;
         end;
      if _ok then output;
      run;
   %end;


%mend;


%macro ds_dupo(in_ds=, out_ds=, key=);
 /* removes duplicates from dataset
  parameters
  in_ds(&key,...)
  key - variables that form key of in_ds
  out_ds(&key,...)

Example:
  data x;
  format date 6. ticker $8. return 6.2;
input date ticker $ return;
  cards;
200101 T .3
200101 T .4
200102 T .2
200101 X .2
200102 X .3
;
%ds_dupo(in_ds=x, out_ds=y, key=DATE TICKER);
proc print data=y;id date ticker;run;
   DATE     TICKER       RETURN
 200101     T              0.30
 200101     X              0.20
 200102     T              0.20
 200102     X              0.30
*/
   %l2ma(list=&key,ma=_k);
   %if &out_ds eq %then %let out_ds = &in_ds;
   proc sort data=&in_ds out=&out_ds;
   by &key;

   data &out_ds;
   set &out_ds;
   by &key;
   if first.&&_k&_k0;
   run;
%mend;

%macro ds_addv(in_ds=, var=, value=, format=$8., out_ds=);
 /* adds a variable to a dataset
  parameters
  in_ds input dataset
  var - name of variable
  value - value of variable
  format - format of variable
  out_ds output dataset. defaults to &in_ds

Example:
 data x;
 format cticker  ticker $8. wgt 6.4;
 input cticker ticker wgt;
 cards;
FMAGX T .5
FMAGX XOM .5
FCNTX T .3
FCNTX R .7
;
%ds_addv(in_ds=x, var=date, value=200101, format=8., out_ds=z);
proc print data=z;run;
OBS     CTICKER      TICKER          WGT         DATE
 1      FMAGX        T             0.5000       200101
 2      FMAGX        XOM           0.5000       200101
 3      FCNTX        T             0.3000       200101
 4      FCNTX        R             0.7000       200101
*/

   %if &out_ds eq %then %let out_ds = &in_ds;
   data &out_ds;
   set &in_ds;
   format &var &format;
   %if %substr(&format,1,1) eq $ %then
      &var = "&value";
   %else
      &var = &value;;
   run;

%mend;

%macro v2vv(in_ds=, var=, out_ds=, prefix=);
 /* creates vars from the contents of a var
  parameters
  in_ds(var1,...,&var,...) &var must be character <= $8.
  var - name of var holdings varnames
  out_ds(var1,var2,...,v1,v2,..)
  where v1,v2,... are the distinct values in the var &var in in_ds.
  contents of vi are 1 or 0 depending
  prefix - prefix to new variable names

Example:
  data x;
```

```
  format ticker $8. atype $6.;
  input ticker $ atype $;
  cards;
XOM STOCKS
_T1 BONDS
_T2 CASH
;
%v2vv(in_ds=x,var=atype,out_ds=y);
proc print data=y;id ticker;run;
 TICKER      ATYPE      BONDS     CASH     STOCKS
 XOM         STOCKS       0         0         1
 _T1         BONDS        1         0         0
 _T2         CASH         0         1         0

%v2vv(in_ds=x,var=atype,out_ds=y,prefix=X_);
proc print data=y;id ticker;run;
 TICKER      ATYPE     X_BONDS    X_CASH    X_STOCKS
 XOM         STOCKS       0          0          1
 _T1         BONDS        1          0          0
 _T2         CASH         0          1          0
*/

   %if &out_ds = %then %let out_ds = &in_ds;

   %ds2ma(in_ds=&in_ds,var=&var,ma=_cv);

   %local i;
   data &out_ds;
   set &in_ds;
   %do i = 1 %to &_cv0;
      %if "&&_cv&i" ne "" %then %do;
         if upcase(&var) eq upcase("&&_cv&i") then
            &prefix.&&_cv&i = 1; else &prefix.&&_cv&i = 0;
      %end;
   %end;
   run;

%mend;

%macro ma_live(in_ds=, in_ma=, out_ma=,nonzero=1);
 /* keeps active variables in a ma
  if in_ma is a macro array of variables in in_tv,
  out_ma will contain the subset for which some row in in_tv
  has a non-zero non-null value
  parameters
  in_ds(ticker,var1,var2,...)
  in_ma - prefix to a macro array
  out_ma - prefix to a macro array - defaults to in_ma
  nonzero - 1 means retain non-zero and non-null variables, 0 means retain non-null va
riables
Example:
  data returns;
input DATE FMAGX FCNTX FXXXX;
cards;
199901 .2 . .
199902 .3 .3 .
299903 .4 .4 .
;
%dsv2ma(in_ds=returns,drop=date,ma=t);
%ma_live(in_ds=returns, in_ma=t,out_ma=s);
%put &s0 &s1 &s2;
2 FCNTX FMAGX
*/

   %if &out_ma eq %then %let out_ma = &in_ma;
   %local i;
   data _a1(keep = _v );
   set &in_ds(keep= %ma2l(ma=&in_ma)) end=eof;
   format _v $8.;
   array _x[&&&in_ma.0];
   array _y[&&&in_ma.0] %ma2l(ma=&in_ma);
   retain _x1 - _x&&&in_ma.0 0;
   do _i = 1 to &&&in_ma.0;
      if _y[_i] ne . %if &nonzero ne 0 %then and _y[_i] ne 0 ; then _x[_i] = 1;
      end;
   if eof then do;
      %do i = 1 %to &&&in_ma.0;
         if _x[&i] ne 0 then do;
            _v = "&&&in_ma&i";
            output;
            end;
      %end;
      end;

   %ds2ma(in_ds=_a1,var=_v,ma=&out_ma);
   %trash(_a1);

%mend;


%macro dsnorm(in_ds=, out_ds=, vars=);
 /* normalizes each of a list of variables. defaults to all variables
  parameters
  in_ds
  out_ds to in_ds
  vars - list of variables to normalize
  the values of each variable are adjusted proportionally to get their sums to add to
1.

  Example:
  data x;
  input n m;
  cards;
.2 .4
.3 .8
;
%dsnorm(in_ds=x);
proc print data=x;run;
 OBS      N         M
 1      0.4     0.33333
 2      0.6     0.66667
*/

   %if &out_ds eq %then %let out_ds = &in_ds;
   %if &vars eq %then %do;
      %dsv2ma(in_ds=&in_ds,ma=_m);
   %end;
   %else %do;
      %l2ma(list=&vars,ma=_m);
   %end;

   proc sql;
   create table _a5 as
      select sum(&&_m1) as _x1   %do i = 2 %to &_m0; ,sum(&&_m&i) as _x&i %end;
   from &in_ds;

   data &out_ds(drop = _i %do i = 1 %to &_m0; _x&i %end;);
   set &in_ds;
   array _z[&_m0] %do i = 1 %to &_m0; &&_m&i %end;;
```

```
   array _x[&_m0] _x1 - _x&_m0;
   retain _x1 - _x&_m0;
   if _n_ eq 1 then set _a5;
   do _i = 1 to &_m0;
      if _x[_i] ne 0 and _x[_i] ne . and _z[_i] ne . then
         _z[_i] = _z[_i] / _x[_i];
      else _z[_i] = .;
      end;
   run;
   %trash(_a5);


%mend;


%macro dsmerge(i=,in_ds=,out_ds=,by=);
/* merge datasets together, one at a time
datasets should be sorted by &by if present
if &i eq 1 &out_ds is created, otherwise it is merged onto
Use this when you get results inside a macro loop and you want to accumulate them hori
zontally
in a dataset.  See also %dsappend, which accumulates results vertically
Example:
assemble some returns(just an example, don't do it this way!):
%let fticks=FMAGX FCNTX FMILX;
%l2ma(list=&fticks,ma=tt);

 %do t=1 %to &tt0;
   %let ticker=&&tt&t;
   %d_fndret(from_dt=20010101, to_dt=20010116, tickers=&ticker, out_dtt=fdtt);
   %dsmerge(i=&t,in_ds=fdtt,out_ds=fnd_dtt);
 %end;

proc print data=fnd_dtt;id date;run;
     DATE       FMAGX         FCNTX         FMILX
 20010108     -0.005436      0.000000      0.004795
 20010109      0.002648     -0.002579     -0.000636
 20010110      0.011670      0.003878      0.035339
 20010111      0.017260      0.004722      0.027983
 20010112     -0.004718     -0.004273      0.001197
 20010116      0.007401      0.007724     -0.001793
*/
   %if &i eq 1 %then %do;
      data &out_ds;
      set &in_ds;
      run;
   %end;
   %else %do;
      data &out_ds;
      merge &out_ds &in_ds;
      %if &by ne %then by &by;;
      run;
   %end;
%mend;

%macro dsappend(i=,in_ds=,out_ds=,same=1);
/* append datasets together, one at a time
if &i eq 1 &out_ds is created, otherwise it is appended to
if &same eq 0 then get union of all variables in all &in_ds, which won't be all the sa
me
Use this when you get results inside a macro loop and you want to accumulate them vert
ically
in a dataset.  See also %dsmerge, which accumulates results horizontally
Example:
assemble some returns(just an example, don't do it this way!):
%let fticks=FMAGX FCNTX FMILX;
%l2ma(list=&fticks,ma=tt);

 %do t=1 %to &tt0;
   %let ticker=&&tt&t;
   %d_fndret(from_dt=20010101, to_dt=20010103, tickers=&ticker, out_dtr=fdtr);
   %dsappend(i=&t,in_ds=fdtr,out_ds=fnd_dtr);
 %end;

 proc sort data=fnd_dtr;
 by date ticker;
 run;
proc print data=fnd_dtr;id date;run;
     DATE     TICKER         RETURN
 20010102     FCNTX        -0.031920
 20010102     FMAGX        -0.031936
 20010102     FMILX        -0.079231
 20010103     FCNTX         0.018694
 20010103     FMAGX         0.055329
 20010103     FMILX         0.093009
*/
   %if &i eq 1 %then %do;
      data &out_ds;
      set &in_ds;
      run;
   %end;
   %else %do;
      %if &same ne 0 %then %do;
         proc append base=&out_ds data=&in_ds force;
         run;
      %end;
      %else %do;
         data &out_ds;
         set &out_ds &in_ds;
         run;
      %end;
   %end;
%mend;

%macro rmlabel(lib=work,ds=);
/* removes labels from dataset
removes labels from all variables in dataset
*/

   %dsv2ma(in_ds=&lib..&ds,ma=_rm);
   %local r;
   proc datasets library=&lib nolist;
   modify &ds;
   label %do r= 1 %to &_rm0;
      &&_rm&r = ' '
      %end;;
   run;

%mend;

%macro dsrange(in_ds=, out_ds=, var=, from=, to=);
/* subsets a dataset by a range of values of a variable
in_ds - input dataset
out_ds - output dataset defaults to &in_ds
var - name of variable
from - from value
to - to value

Example:
 data x;
```

```
 format ticker $8.;
 input date ticker $ return;
 cards;
200101 T .3
200102 T .4
200103 T .2
200101 X .2
200102 X .3
200103 X .2
200104 X .3
;
%dsrange(in_ds=x,out_ds=y,var=date,from=200102,to=200103);
 proc print data=y; run;
OBS     TICKER       DATE     RETURN
 1      T           200102      0.4
 2      T           200103      0.2
 3      X           200102      0.3
 4      X           200103      0.2
*/
%if &out_ds eq %then %let out_ds=&in_ds;

data &out_ds;
   set &in_ds(where=(&var >= &from and &var <= &to));
   run;

%mend;



%macro appendym(from_dt=, to_dt=, libdir=, in_ds=, out_ds=, add_dt=0, same=1);
/* append together datasets in date-separated libraries
If you have multiple datasets of the same name in different libraries, where the
last part of the library directory path is the date,  then you can use this macro
to assemble them all into a single dataset.
parameters
from_dt - yyyymm from date
to_dt   - yyyymm to date
libdir  - each libname="&libdir/YYYYMM"
in_ds - name of datasets
out_ds output dataset
add_dt - nonzero means add date=yyyymm to merged dataset
if &same eq 0 then get union of all variables in all &in_ds, which won't be all the sa
me

Example:
 libname x200011 '/home/bvick/test/200011';
 libname x200012 '/home/bvick/test/200012';
data x200011.a;
format ticker $8.;
input ticker wgt;
cards;
T .11
X .12
;
data x200012.a;
format ticker $8.;
input ticker wgt;
cards;
T .21
X .22
;
run;
%appendym(from_dt=200011,to_dt=200012,libdir=/home/bvick/test, in_ds=a, out_ds=z,add_d
t=1);
 proc print data=z; run;
OBS        DATE    TICKER       WGT
1       200011     T          0.11
2       200011     X          0.12
3       200012     T          0.21
4       200012     X          0.22
*/

   %local m got;
   %let got = 0;
   %ym2ma(from_dt = &from_dt, to_dt=&to_dt, ma=_mm);
   %do m = 1 %to &_mm0;
      libname _mm "&libdir/&&_mm&m";
      %if %sysfunc(exist(_mm.&in_ds)) %then %do;
         %let got = %eval(&got + 1);
         %if &add_dt ne 0 %then %do;
            data _mm;
            format date 8.;
            set _mm.&in_ds;
            date = &&_mm&m;
            run;
            %dsappend(in_ds=_mm,out_ds=&out_ds,i=&m,same=&same);
         %end;
         %else %do;
            %dsappend(in_ds=_mm.&in_ds,out_ds=&out_ds,i=&m,same=&same);
         %end;
      %end;
   %end;
%mend;


%macro mkdivvy(in_ds=,vars=TICKER,filename=,program=);
/* make a runfile for parallel sas job runs via sasdivvy
in_ds(&VARS) - dataset containing a record for each run and variables &VARS
vars - relevant variable names. Each variable gets set to an environment variable on c
all to &program
filename - name of runfile
the output log file are in cwd, with names that are _ separated concatenations of vari
able values
Example:
 data funds;
 format ticker $8.;
 input ticker $;
 cards;
FMAGX
FCONX
FMILX
;
run;
%mkdivvy(in_ds=funds,filename=foo,program=/home/bvick/test/analyze);
x 'cat foo';
-set TICKER FMAGX -log  _FMAGX.log /home/bvick/test/analyze
-set TICKER FCONX -log  _FCONX.log /home/bvick/test/analyze
-set TICKER FMILX -log  _FMILX.log /home/bvick/test/analyze
To run, type
/project/quantitative1/fppadmin/projects/utilities/sasdivvy -from 1 -to 3 -v  foo
*/
   %local v;
   %l2ma(list=&vars,ma=vv);

   filename x "&filename";
   data _null_;
   format x y $150.;
   set &in_ds(keep=&vars);
   file x;
   x = '';
```

```
   %do v = 1 %to &vv0;
      x = trim(x) || " -set &&VV&V " || trim(left(&&&VV&V));
      y = trim(y) || "_" || trim(left(&&VV&V));
      y = translate(y,'_','/');
   %end;
   x = trim(x) || " -log " || trim(y) || ".log -print " || trim(y) || ".lst &program";
   put x;
   run;

%mend;

%macro csv2ds(in_file=, out_ds=, delim=,vars= ,formats=);
/* import a csv file into a dataset
in_file - name of csv file (including .csv or other extension and directory if not cwd
)
out_ds - name of output dataset
delim - column separater in &in_file, defaults to ,
vars - list of variable names in the order they appear in &in_file.
       Omit to indicate that the first input line contains valid variable names
formats - list of formats for variable.
      If omitted, all variables are assumed to be numeric and BEST. is used

Example:
x 'cat x.csv';
T,X,Y
A,11,12
B,21,22
%csv2ds(in_file=x.csv, out_ds=x, formats=$8. best. best.);
 proc print data=x; run;
OBS    T            X     Y
 1     A           11    12
 2     B           21    22
*/

   %local i;
   filename csvin "&in_file";

   %if %length(&delim) eq 0 %then %let delim = ,;

   %if %length(&vars) eq 0 %then %do;

      data _var;
      infile csvin obs=1 delimiter= "&delim" missover dsd lrecl=10000;
      length var1-var100 $8;
      input var1-var100 $;
      run;

      data _null_;
      set _var(obs=1);
      format f $8.;
      array x[100] var1 - var100;
      c=0;
      do i=1 to 100;
         if (x[i] ne "") then do;
            x[i] = upcase(x[i]);
            f = compress("_var"||put(i,4.));
            call symput(f,x[i]);
            c = c + 1;
         end;
      end;
      call symput("_var0",put(c,4.));
      run;
   %end;
   %else %do;
      %l2ma(list=&vars,ma=_var);

   %end;

   %if %length(&formats) ne 0 %then %do;
      %l2ma(list=&formats,ma=_fmt);
   %end;
   %else %do;
      %do i = 1 %to &_var0;
         %let _fmt&i = BEST.;
      %end;
      %let _fmt0 = &_var0;
   %end;
   %do i = 1 %to &_fmt0;
      %if "%substr(&&_fmt&i,1,1)" eq "$" %then %let _inf&i = $;
      %else %let _inf&i =;
   %end;

   data &out_ds;
   format
      %do i = 1 %to &_var0;
         &&_var&i &&_fmt&i
      %end;;
   infile csvin delimiter= "&delim" missover dsd %if %length(&vars) eq 0 %then firstob
s=2; lrecl=10000;
   input
      %do i = 1 %to &_var0;
         &&_var&i &&_inf&i
      %end;;
   run;

%mend;

/* internal use only */ %macro out_col( varN, varF, varT, isCont, isComa );
   %if ( "&varT" eq "str" ) %then %do;

      if ( index(&varN, '"') ) then do;
         &varN=tranwrd( &varN, """", """""" );
         end;

      if index(trim(&varN), ',') then do;
         _efitmp_ = '"'||trim(left(&varN))||'"';
         end;
      else _efitmp_ = '"'||trim(left(&varN))||'"';

      put  %if ("&isComa" eq "yes") %then "," ; _efitmp_ $ +(-1)
         %if ("&isCont" eq "yes") %then @ ; ;

   %end;
   %else %do;

      /*-- Process numeric variable for imbedded delimiter or quotes --*/
      if &varN > .z then do;
         _efitmp_ = left(put( &varN, &varF));
         if index(trim(_efitmp_), ',') or index(_efitmp_, '"') then
            do;
               _efitmp_ = '"'||_efitmp_||'"';
               end;
         put  %if ("&isComa" eq "yes") %then ","; _efitmp_ $ +(-1)
            %if ("&isCont" eq "yes") %then @ ; ;
         end;
            else put %if ("&isComa" eq "yes") %then "," ;
      %if ("&isCont" eq "yes") %then  @  ; ; /* missing */

   %end;
```

```
%mend;

%macro ds2csv(in_ds=, out_file=, head=1, label=0);
/* make csv file from dataset
in_ds() - dataset
out_file - file name
head - nonzero means include headings
label - nonzero means include labels

Example:
 data x;
 format t $8.;
 input t x y;
 cards;
A 11 12
B 21 22
;
run;
%ds2csv(in_ds=x, out_file=x.csv);
 x 'cat x.csv';
T,X,Y
"A",11,12
"B",21,22
*/



    proc datasets nolist;
        contents data=&in_ds out=_fmtds noprint;
    run;

    proc sort data=_fmtds;
        by varnum;

%let XNLAST=0;

    data _null_;
        length mname $10;
        set _fmtds end=eflag;
        mname="XN"||trim(left(put(_n_,10.)));
        call symput(mname,trim(left(name)));
        mname="XLB"||trim(left(put(_n_,10.)));
        call symput(mname,trim(label));
        mname="XLN"||trim(left(put(_n_,10.)));
        call symput(mname,trim(left(put(length,10.))));

        if ( type eq 2 ) then do;
            mname="XF"||trim(left(put(_n_,10.)));
            call symput(mname,"$"||trim(left(put(length,10.)))||".");
            mname="XD"||trim(left(put(_n_,10.)));
            call symput(mname,"str");
            mname="XST"||trim(left(put(_n_,10.)));
            call symput(mname,"$");
        end;
        else do;
            mname="XF"||trim(left(put(_n_,10.)));
            if ( format ne "" ) then do;
                call symput(mname,
                            trim(left(format))||
                            trim(left(put(formatl,10.)))||"."||
                            trim(left(put(formatd,10.))));
            end;
            else do;
                call symput(mname,"best12.");
            end;
            mname="XD"||trim(left(put(_n_,10.)));
            call symput(mname,"num");
            mname="XST"||trim(left(put(_n_,10.)));
            call symput(mname," ");

        end;

        if ( eflag ) then do;
            call symput("XNLAST",trim(left(put(_n_,10.))));
        end;
    run;


    data _null_;
    set  &in_ds  end=EFIEOD;
    %let _EFIERR_ = 0;      /* clear ERROR detection macro variable */
    %let _EFIREC_ = 0;      /* clear export record count macro variable */
    file "&out_File"   lrecl=32767;

       %do j=1 %to &XNLAST;
           format &&XN&J &&XF&J;
       %end;
       /* temporary variable for quote and imbedded delimiter handling */
       format _efitmp_ $200. ;
    if _n_ = 1 then do;
       %if &head ne 0 %then %do;
          put "&XN1" %do j=2 %to &XNLAST; "," "&&XN&J" %end; ;
       %end;
       %if &label ne 0 %then %do;
          put """&XLB1""" %do j=2 %to &XNLAST; "," """&&XLB&J""" %end; ;
       %end;
    end;

    do;
       EFIOUT + 1;

       %if %eval(&xnlast eq 1) %then %do;

          %out_col( &XN1, &XF1, &XD1 , no, no);
       %end;
       %else %do;
          %out_col( &XN1, &XF1, &XD1 , yes, no);

          %do j = 2 %to %eval(&XNLAST-1);

             %out_col( &&XN&J, &&XF&J, &&XD&J, yes, yes );

          %end;

          %out_col( &&XN&XNLAST, &&XF&XNLAST, &&XD&XNLAST , no, yes);
       %end;

    end;

    If _ERROR_ then         /* ERROR detection */
       call symput('_EFIERR_',1);
    If EFIEOD then
       call symput('_EFIREC_',EFIOUT);
    run;

%mend;


%macro ds2chds(in_ds=, out_ds=,vars=);
```

```
/* converts all numeric variables in a dataset to character variables based on format
in_ds - input dataset
out_ds - output dataset defaults to &in_ds
vars - optional list of variables to keep(defaults to all)
each character variable is moved unchanged
numeric variables are converted to character variables using whatever formats they hav
e in &in_ds
*/
   proc contents data=&in_ds %if "&vars" ne "" %then (keep=&vars);
      out=_m1 noprint;
   run;
   proc sort data=_m1;
   by varnum;
   run;

   data _m1(keep=name fmt label type );
   length fmt $ 15 f $ 2;
   set _m1;
   if formatd ne 0 then f=put(formatd,2.);
   else f='';
   fmt=compress(format||put(format1,4.)||'.'||f);
   if label = '' then label=name;
   run;

   %ds2ma(in_ds=_m1,var=name  ,ma=_dsn,uniq=0);
   %ds2ma(in_ds=_m1,var=fmt,ma=_dsf,uniq=0);
   %ds2ma(in_ds=_m1,var=label ,ma=_dsl,uniq=0);
   %ds2ma(in_ds=_m1,var=type  ,ma=_dst,uniq=0);

   %local i ;

   data &out_ds(keep=%do i=1 %to &_dsn0; &&_dsn&i %end;);
   label %do i=1 %to &_dsn0; &&_dsn&i="&&_dsl&i" %end;;
   set &in_ds( %if "&vars" ne "" %then keep=&vars; rename=(%do i=1 %to &_dsn0; %if &&_
dst&i ne 2 %then &&_dsn&i=__x&i; %end;));
   %do i = 1 %to &_dsn0;
      %if &&_dst&i ne 2 %then &&_dsn&i = put(__x&i, &&_dsf&i);;
   %end;
   run;

%mend;


%macro ds2pct(in_ds=, out_ds=);
/* convert variables to percentages
 multiply all non-date numeric variables by 100 and change format to 6.2;
 if there is a "date" variable, it is left unchanged
Example:
 data x;
 format date 6. FMAGX FCNTX 6.4;
 input date FMAGX FCNTX;
 cards ;
200101 -0.04 0.05
200102 0.30  0.35
;
%ds2pct(in_ds=x);
 proc print data=x; run;
OBS      DATE     FMAGX      FCNTX
 1      200101     -4.00      5.00
 2      200102     30.00     35.00

*/

   proc datasets nolist;
        contents data=&in_ds out=_fmtds noprint;
   run;
   data _fmtds(keep=name);
   set _fmtds(where=(type eq 1 and name ne 'DATE'));
   run;

   %ds2ma(in_ds=_fmtds,var=name,ma=__ds);
   %if &out_ds eq %then %let out_ds= &in_ds;
   %local i;

   data &out_ds;
   set &in_ds;
   format %do i = 1 %to &__ds0;  &&__ds&i %end; 6.2;
   %do i = 1 %to &__ds0;  &&__ds&i= 100*&&__ds&i; %end;
   run;

%mend;

%macro trashlib(lib=);
/* delete all datasets in a library
*/
   %lib2ma(lib=&lib,ma=gds);
   %let _gds = %ma2l(ma=gds);
   %if &_gds ne %then %trash(&_gds,&lib);

%mend;

%macro truncout(in_ds=,out_ds=,var=,pct=,minval=,maxval=);
/* truncate outlier values
var - list of variables
if &pct > 50, then set any value of &var over  the &pct perctile value to the &pct per
centile value
if &pct < 50, then set any value of &var undertthe &pct perctile value to the &pct per
centile value
(currently only supports 1 variable in pct mode)
if &pct blank, then use &minval and &maxval to control values
*/

   %l2ma(list=&var,ma=_v);

   %if &out_ds eq %then %let out_ds=&in_ds;

   %if &pct ne %then %do;
      proc univariate data=&in_ds;
      output out=pct pctlpts=&pct pctlpre=_P;
      var &var;
      run;
      %let _pct = ;

      proc sql noprint;
      select _P&pct into :_pct
         from pct;
      %if "&_pct" ne "" %then %do;

         data &out_ds;
         set &in_ds;
         %if %sysevalf(&pct > 50) %then %do;
            if &var > &_pct then &var = &_pct;
         %end;
         %else %do;
            if &var < &_pct then &var = &_pct;
         %end;
      %end;
      %else %do;
         data &out_ds;
         set &in_ds;
```

```
        run;
      %end;
   %end;
   %else %do;
      %local i;
      data &out_ds;
      set &in_ds;
      %do i = 1 %to &_v0;
         %if &minval ne %then if &&_v&i < &minval then &&_v&i = &minval;;
         %if &maxval ne %then if &&_v&i > &maxval then &&_v&i = &maxval;;
      %end;
      run;
   %end;

   run;

%mend;

%macro put1(in_ds);
/* put first row of dataset vertically */
   data _null_;
   set &in_ds(obs=1);
   array _x _character_;
   do over _x;
      put _x= ;
      end;
   put;
   array _y _numeric_;
   do over _y;
      put _y= ;
      end;
   run;

%mend;
```

```
/*
    $Author: xen $

    $Date: 1999/01/06 16:44:13 $

    $Source: /project/watermdl/waterStreet/projects/utilities/import_csv.macro,v $

    $Log: import_csv.macro,v $
# Revision 1.4  1999/01/06  16:44:13  xen
# Y2k compliance.
#
# Revision 1.3  1998/03/27  17:57:46  rhua
# check in again
#
# Revision 1.1  1998/01/21  16:50:21  rhua
# Initial revision
#
    Revision 1.1.1.1  1997/12/08 19:48:18  rhua
    Check in sources the first time


*/

/*****************************************************************
*
*   IMPORT to CSV Format
*
*   by Ronald Hua, 07/23/97
*
*****************************************************************/


**                                                               ;
* MACROS ---------------------------------------------------------;
*                                                               ;



%macro imp_csv( fileN, outDs, skipN, delm );

    filename csvin "&fileN";
    %let maxVars=500;

    data _tmp;
        infile csvin  delimiter = "&delm" MISSOVER DSD firstobs=%eval(&skipN+1) LRECL=
10000;
        length var1-var&maxVars $30;

        input var1-var&maxVars $ ;

        if _n_ gt 4 then stop;
    run;

    proc transpose data=_tmp out=_tmp01;
         var var1-var&maxVars;
    run;

    data _null_;
        length mname $10;
        set _tmp01;

        if (trim(col1) eq "") then do;
            call symput("MAXINP",trim(left(put(_n_-1,10.))));
            stop;
        end;

        mname="V"||trim(left(put(_n_,10.)));
        call symput(mname,trim(col1));
        mname="T"||trim(left(put(_n_,10.)));
        call symput(mname,trim(col2));
        mname="L"||trim(left(put(_n_,10.)));
        call symput(mname,trim(col3));
        mname="B"||trim(left(put(_n_,10.)));
        call symput(mname,trim(col4));
    run;

    %do j=1 %to &MAXINP;
        %put &J > &&V&J &&T&J &&L&J &&B&J;
    %end;

    data &outDs;
        infile csvin  delimiter = "&delm" MISSOVER DSD firstobs=%eval(&skipN+5) LRECL=
10000;
        length %do j=1 %to &MAXINP;
                %if ( "&&T&J" eq "$" ) %then &&V&J &&T&J.&&L&J ; %end; ;

        input %do j=1 %to &MAXINP; &&V&J &&T&J  %end; ;

        %*if &&V&1 ne "" ;

        label %do j=1 %to &MAXINP; &&V&J = "&&B&J" %end; ;
    run;

    proc contents data=&outDs;
    run;
/*
    proc print data=&outDs;
    run;
*/


%mend;
```

keyvar.macro


```
/*  perform operations on (key, variable1, ...) datasets
include $FPPCODE/utilities/keyvar.macro
depends on
$FPPCODE/macroman.macro
$FPPCODE/dsutil.macro

key can usually be multiple variables.  typically the dataset should be sorted or inde
xed by the key
*/

%macro kvv2kkv(in_ds=, out_ds=, key=, vars=, out_var=, out_key=, keep0=0);
 /* conversion: key>key, var-names>&out_key, var-values>out_var
  parameters
  in_ds(&key,var1,...)
  key - key variable name(s)
  vars - variable name(s) for conversion
  out_var - output variable name
  out_key - name of additional key variable
  out_ds(&key,&out_key,&out_var)
  */

   data &out_ds(keep = &out_key &key &out_var);
   format &out_key $8.;
   set &in_ds(keep = &key &vars);
   array _x &vars;
   do over _x;
      call vname(_x,&out_key);
      &out_var = _x;
      %if &keep0 eq 0 %then if &out_var ne . and &out_var ne 0 then; output;
      end;
   run;

   proc sort data=&out_ds;
   by &out_key &key;
   run;

%mend;


%macro kvvw2kkw(in_ds=, out_ds=, key=, vars=, out_var=, out_key=, keep0=0);
/* conversion: key>key, var-names>keym, var-values>out_var
  parameters
  in_ds(&key,var1,...)
  key - key variable name(s)
  vars - variable name(s) for conversion
  out_var - output variable name
  out_key - name of additional key variable
  out_ds(&key,&out_key,&out_var)
  */

   data &out_ds(keep = &out_key &key &out_var);
   format &out_key $8.;
   set &in_ds(keep = &key &vars);
   array _x &vars;
   do over _x;
      call vname(_x,&out_key);
      &out_var = _x;
      %if &keep0 eq 0 %then if &out_var ne . and &out_var ne 0 then; output;
      end;
   run;

   proc sort data=&out_ds;
   by &out_key &key;
   run;

%mend;

%macro kv_kv(in_ds1=, in_ds2=, out_ds=, key=, col=OR, row=OR, sort=0, drop1=, drop2=);
 /* merge two keyed datasets
  parameters
  in_ds1(&key,var,...)
  in_ds2(&key,var,...)
  out_ds(&key,var...)
  key - variable(s) that form key to datasets
  col - boolean rule for selecting non-key output columns
  row - boolean rule for selecting output rows
  sort - indicates in_ds's need to be sorted
  drop1 - variables to omit from in_ds1
  drop2 - variables to omit from in_ds2
*/
   %if (%sysfunc(exist(&in_ds1)) eq 0) %then %do;
      data &in_ds1;
      set &in_ds2(obs=0 keep=&key);
      run;
   %end;
   %if (%sysfunc(exist(&in_ds2)) eq 0) %then %do;
      data &in_ds2;
      set &in_ds1(obs=0 keep=&key);
      run;
   %end;


   %dsv2ma(in_ds=&in_ds1,drop=&key &drop1,ma=_mma);
   %dsv2ma(in_ds=&in_ds2,drop=&key &drop2,ma=_mmb);
   %ma_ma(maa=_mmb,mab=_mma,mac=_mmc,op=and); /* get var's in both b and a */
   %local i;
   %if &sort ne 0 %then %do;
      proc sort data=&in_ds1;
      by &key;
      proc sort data=&in_ds2;
      by &key;
      run;
   %end;

   /* get overlapping character, numeric variables */

   %if &_mmc0 ne 0 %then %do;
      data _d1n(keep=var) _d1c(keep=var);
      set &in_ds2(keep=%do i = 1 %to &_mmc0; &&_mmc&i %end; obs=1) ;
      array _n(*) _numeric_;
      array _c(*) _character_;
      length var $8;
      do _i = 1 to dim(_n);
         call vname(_n(_i),var);
         output _d1n;
         end;
      do _i = 1 to dim(_c);
         call vname(_c(_i),var);
         output _d1c;
         end;
      run;
      %ds2ma(in_ds=_d1n,var=var,ma=_dn);
      %ds2ma(in_ds=_d1c,var=var,ma=_dc);
      %trash(_d1n _d1c);
   %end;
   %else %do;
      %let _dn0 = 0;
      %let _dc0 = 0;
   %end;
```

```
   %ma_ma(maa=_mma,mab=_mmb,mac=_mmo,op=&col);
   data &out_ds(drop = %do i = 1 %to &_dn0; _dn&i %end; %do i = 1 %to &_dc0; _dc&i %en
d;
                keep = &key %ma2l(ma=_mmo) );
   merge &in_ds1(in=ina keep=&key %ma2l(ma=_mma))
      &in_ds2(in=inb keep= &key %ma2l(ma=_mmb)
              rename=( %do i = 1 %to &_dn0; &&_dn&i=_dn&i %end; %do i = 1 %to &_dc0; &
&_dc&i=_dc&i %end; )
              );
   by &key;
   if ina &row inb;
   %do i = 1 %to &_dn0;
      if &&_dn&i eq . then &&_dn&i = _dn&i;
   %end;
   %do i = 1 %to &_dc0;
      if &&_dc&i eq '' then &&_dc&i = _dc&i;
   %end;
   run;

%mend;



%macro kvx_kv(in_ds1=, key=ticker, out_ds=);
 /* horiz/vertical merge of n datasets into a single one.
  parameters
  dsx - list of datasets to merge together. first on list gets precendence
  var - name of variable that forms the key
  out_ds - output dataset.      shouldn't be one of the input ones
  */
   %l2ma(list=&in_ds1,ma=_dsx);
   %local i;
   %kv_kv(in_ds1=&_dsx1,in_ds2=&_dsx2,key=&key,out_ds=&out_ds);
   %do i = 3 %to &_dsx0;
      %kv_kv(in_ds1=&out_ds,in_ds2=&&_dsx&i,key=&key,out_ds=&out_ds);
   %end;

%mend;



%macro v_deref(in_ds=, in_ref=, var=, out_ds=, drop=0, outsort=, keep=);
 /* dereference a variable, subject to a lookup table
  parameters
  in_ds(ticker, var1,...,&var,...)
  in_ref(&var, rvar1,...rvarn) sorted by &var
  var - name of var in in_ref
  out_ds(ticker,var1,...,&var,...,rvar1,...rvarn) if drop=0, without &var if drop=1
  drop 1 means drop &var, default 0 means keep &var
  outsort - key variable(s) to sort output by
  keep - variables in &in_ref in addition to &var to keep (defaults to all)
  */

   %if &out_ds eq %then %let out_ds = &in_ds;

   proc sort data=&in_ds out=&out_ds;
   by &var;

   data &out_ds %if &drop=1 %then %do; (drop=&var) %end;;
   merge &out_ds(in=ina) &in_ref %if &keep ne %then  %do; ( keep= &var &keep )%end;;
   by &var;
   if ina;
   run;

   %if &outsort ne %then %do;
      proc sort data=&out_ds;
      by &outsort;
      run;
   %end;

%mend;



%macro v_replac(in_ds=, in_ref=, var=, repvar=, out_ds=,refvar=);
 /* dereference a variable, replacing var with replacement var
  parameters
  in_ds(ticker, var1,...,&var,...)
  in_ref(&var, &repvar,...)
  var - name of var in  in_ref
  repvar - name of var from in_ref to use as replacement
  out_ds(ticker,var1,...,&var,...)
  refvar - name of variable in refds that has &var values (defaults to &var)
  */
   %if &out_ds eq %then %let out_ds = &in_ds;
   %if &refvar eq %then %let refvar = &var;

   proc sort data=&in_ds out=_a1;
   by &var;
   proc sort data=&in_ref(keep=&refvar &repvar) out=_a2;
   by &refvar;

   data &out_ds (drop=&var rename=(__x=&var));
   merge _a1(in=ina) _a2(rename=(&repvar = __x &refvar=&var));
   by &var;
   if ina;
   run;
   %trash(_a1 _a2);

%mend;



%macro vvnorm(in_ds=, vars=, vards=, var=, default=, out_ds=);
 /* horizontally normalizes a set of vars
  If default is there, don't normalize, rather
  stick the difference in that var(which would typically be "UNASSIGNED".
  parameters
  in_ds(var1,...varn)
  choose var | vards
  vars - list of variables
  vards - dataset containing variable names in variable &var
  default - place to put difference between current weights and 1.
  if absent, normalize
  out_ds(var1,...varn) defaults to in_ds
  */
   %local i;
   %if &out_ds eq %then %let out_ds = &in_ds;

   %if &vars ne %then %l2ma(list=&vars,ma=_cv);
   %else %ds2ma(in_ds=&vards,var=&var,ma=_cv);

   data &out_ds(drop=_s);
   set &in_ds;
   _s = 0;
   array _x %do i = 1 %to &_cv0; &&_cv&i %end;;
   do over _x;
      if _x = . then _x = 0;
      _s = _s + _x;
      end;
```

```
   %if &default ne %then %do;
      if &default = . then &default = 0;
     &default = &default + 1 - _s;
   %end;
   %else %do;
      do over _x;
         if _s = 0 then _x = .;
         else _x = _x / _s;
         end;
   %end;
   run;


%mend;


%macro kkvnorm(in_ds=,  out_ds=, default=, vars=, key1=, key2=, key3=);
 /* normalize a two-key dataset
  parameters
  in_ds(key1,key2,&vars)
  out_ds(key1,key2,&vars)
  default - &key2 name for part not adding to 1
  */
   %local i primkeys lastkey allkeys;
   %if &out_ds eq %then %let out_ds = &in_ds;
   %if &key3 eq %then %do;
      %let primkeys = &key1;
      %let allkeys = a.&key1, a.&key2;
      %let lastkey = &key2;
   %end;
   %else %do;
      %let primkeys = &key1, &key2;
      %let allkeys = a.&key1, a.&key2, a.&key3;
      %let lastkey = &key3;
   %end;



   %dsv2ma(in_ds=&in_ds,drop=&vars &key1 &key2 &key3,ma=_oth);
   %let _other = %ma2l(ma=_oth);
   %let _otherc = %l2cl(&_other);

   %l2ma(list=&vars,ma=_kk);

   proc sql;
   create table _kk3 as
      select &primkeys %do i=1 %to &_kk0; ,sum(&&_kk&i) as &&_kk&i %end;
   from &in_ds
      group by &primkeys;


   %if &default eq %then %do;
      proc sql;
      create table &out_ds as select
         &allkeys %if "&_otherc" ne "" %then ,&_otherc;  %do i=1 %to &_kk0; , a.&&_kk&
i/s.&&_kk&i as &&_kk&i %end;
      from _kk3 s, &in_ds a
         where s.&key1 = a.&key1
         %if &key3 ne %then and s.&key2 = a.&key2;
      order by &allkeys;

   %end;
   %else %do;
      data _kk3;
      set _kk3;
      format &lastkey $8.;
      %do i=1 %to &_kk0;
         &&_kk&i=1-&&_kk&i;
      %end;
      &lastkey = "&default";

      data &out_ds;
      set &in_ds _kk3;
      run;
      proc sort data=&out_ds;
      by &key1 &key2 &key3;
   %end;

   %trash(_kk3);

%mend;



%macro kvnorm(in_ds=, out_ds=, vars=, key=ticker);
 /* normalizes each of a list of variables. defaults to all non-key variables
  parameters
  in_ds
  out_ds defaults to in_ds
  vars - list of variables to normalize
  key - key variables
  */

   %if &out_ds eq %then %let out_ds = &in_ds;
   %l2ma(list=&key,ma=_k);

   %if &vars eq %then %do;
      %dsv2ma(in_ds=&in_ds,drop=&key,ma=_m);
   %end;
   %else %do;
      %l2ma(list=&vars,ma=_m);
   %end;

   %local i;

   proc sql;
   create table _a5 as
      select %do i = 1 %to &_k0; &&_k&i, %end; sum(&&_m1) as _x1   %do i = 2 %to &_m0;
,sum(&&_m&i) as _x&i %end;
   from &in_ds
      group by &&_k1 %do i = 2 %to &_k0; ,&&_k&i %end;;

   data &out_ds(drop= _i %do i = 1 %to &_m0; _x&i %end;);
   merge _a5 &in_ds;
   by %do i = 1 %to &_k0; &&_k&i %end;;
   array _z[&_m0] %do i = 1 %to &_m0; &&_m&i %end;;
   array _x[&_m0] _x1 - _x&_m0;
   do _i = 1 to &_m0;
      if _x[_i] ne 0 and _x[_i] ne . and _z[_i] ne . then
         _z[_i] = _z[_i] / _x[_i];
      else _z[_i] = .;
      end;
   run;
   %trash(_a5);

%mend;

%macro kv_mbr(in_ds=, in_keyds=, key=, yes_ds=, no_ds=, sort=0 );
/* makes 2 datasets based on presence of &in_ds records in &in_keyds
in_ds(key,...) input dataset to be split
```

```
in_keyds(key,...) keys that determine output (will be sorted )
key - list of key variables
yes_ds(key,...) records from &in_ds whose key variables are in &in_keyds
no_ds(key,...)  records from &in_ds whose key variables are not in &in_keyds
sort - 1 means &in_ds is not sorted by key
*/

   %let _kvm = %l2cl(&key);

   %if &sort ne 0 %then %do;
      proc sort data=&in_ds out=_kvm1;
      by &key;
      %let in_ds = _kvm1;
   %end;

   proc sort data=&in_keyds;
   by &key;

   data &yes_ds &no_ds;
   merge &in_ds(in=ina) &in_keyds(keep = &key in=inb);
   by &key;
   if ina;
   if inb then output &yes_ds;
   else output &no_ds;
   run;

   %if &sort ne 0 %then %trash(_kvm1);   ;

%mend;


%macro set_repl(base_ds=, in_ds=, out_ds=, var= );
/* replace all observations in &base_ds having &var in &in_ds
&out_ds gets all observations in &in_ds plus those in &base_ds that don't have &var va
lues in &in_ds
var - list of variables.
base_ds(&var,...) base dataset
in_ds(&var,...) new dataset
out_ds(&var,...) output dataset. defaults to base_ds
Note that the out_ds will not be sorted
&in_ds and &out_ds must both have &var but otherwise need not have same variables.
*/
   %if &out_ds eq %then %let out_ds= &base_ds;
   %let _srcl = %l2cl(&var);

   proc sql noprint;
   create table _keys as
   select distinct &_srcl
      from &in_ds
      order by &_srcl;

   proc sort data=&base_ds out=&out_ds;
   by &var;
   run;

   data &out_ds;
   merge &out_ds _keys(in=_ink);
   by &var;
   if not _ink;
   run;

   data &out_ds;
   set &out_ds &in_ds;
   run;
%mend;
```

```
%macro mkproxy(in_ds=, keyvar=, sigvars=, proxyvar=, out_ds=, map_ds=);
/* make dataset that has proxies for input dataset
in_ds(&keyvar,&sigvars,...) input dataset
keyvar- variable that is indentifier for set
sigvars - signature variables i.e. variables whose combined values form signature.
out_ds(same as &in_ds) - output dataset: just the proxies
map_ds(same as &in_ds plus &proxyvar) all the &in_ds records with the &proxyvar variab
le added
This eliminates duplicate sets of data by choosing one set as a proxy.
For example, to eliminate identical portfolios from a ctw-type dataset,
%mkproxy(in_ds=x_ctw, keyvar=CTICKER, sigvars=TICKER WGT, out_ds=y_ctw, map_ds=xy_map,
proxyvar=prox);
in this case, if funds A and B have same holdings,
xy_map will look like this:
cticker ticker wgt prox
A       T1     .5   A
A       T2     .5   A
B       T1     .5   A
B       T2     .5   A
and y_ctw will omit the records from x_ctw where cticker = 'B'

*/

   %l2ma(list=&sigvars,ma=_s);

   proc sort data=&in_ds out=_sigsin;
   by &sigvars;
   run;

   data _sigs;
   set _sigsin(keep=&sigvars);
   by &sigvars;
   if first.&&_s&_s0;
   run;

   data _sigs;
   set _sigs end=eof;
   _n = _n_;
   if eof then call symput("_sigs", compress(put(_n,5.)));
   run;

   data _sigds;
   merge _sigsin _sigs;
   by &sigvars;
   run;

   proc sort data=_sigds;
   by &keyvar _n;
   run;

   data _signa;
   set _sigds;
   length _sig $&_sigs.;
   retain _sig;
   by &keyvar;
   if first.&keyvar then do;
      _sig=repeat('0',&_sigs);
   end;
   substr(_sig,_n) = '1';
   if last.&keyvar then do;
      output;
   end;
   run;
```

```
   proc sort data=_signa;
   by _sig &keyvar;
   run;

   data _signa(keep=&keyvar &proxyvar);
   set _signa;
   by _sig;
   retain &proxyvar;
   if first._sig then &proxyvar=&keyvar;
   run;

   proc sort data=_signa;
   by &keyvar;
   run;

   data &map_ds;
   merge &in_ds _signa;
   by &keyvar;
   run;

   %if &out_ds ne %then %do;
      data &out_ds;
      set &map_ds(where=(&keyvar = &proxyvar));
      run;

   %end;

%mend;

%macro zoomin(in_ds=,out_ds=,vars=,keyvar=,keyval=,pct=);
/* zoom in to a neighborhood of a certain point in a dataset
in_ds (&keyvar,&vars,...)
out_ds(&keyvar,&vars,...) defaults to &in_ds
vars - list of variables that should be zoomed
keyvar,keyval,pct : Use the observation where &keyvar = "&keyval" as the focal point:
For each variable in &vars, only include in &out_ds those observations that are within
 &pct percent
of the focal point value.
*/
   %local i v _zfrom _zto _zvvalue;
   %l2ma(list=&vars, ma=_zv);
   %if &out_ds eq %then %let out_ds=&in_ds;

   %do i= 1 %to &_zv0;
      %let v = &&_zv&i;

      proc sql noprint;
      select &v
         into :_zvvalue
         from &in_ds
         where &keyvar eq "&keyval";
      proc sql noprint;
      select &_zvvalue - (max(&v) - min(&v))*(&pct/100) ,  &_zvvalue + (max(&v) - min(
&v))*(&pct/100)
         into :_zfrom, :_zto
         from &in_ds;
      data &out_ds;
      set %if &i eq 1 %then &in_ds; %else &out_ds; (where=(&v >= &_zfrom and &v <= &_z
to));
      run;
   %end;

%mend;
```

```
/* macro variable manipulators
%include $FPPCODE/utilities/macros.macro

"ma" refers to a macro array prefix,
for example a1,a2,...  by convention, a0 contains the number of elements in a.
"list" refers to a list of values, normally blank-separated
*/

%macro ds2l(in_ds=, var=, uniq=1, list=, where= );
 /* dataset variable values to a list
  parameters
  in_ds(&var,...) - input dataset
  var - name of variable
  uniq - nonzero to produce sorted unique list. Set to 0 to retain dataset order
  list - name of macro to put list in
  where - optional proc sql-type where clause for filtering in_ds

  Example:
  data x;
  format a b $8.;
  input a $ b $;
  cards;
aa ba
ab bb
ac bc
;
%ds2l(in_ds=x, var=a, list=l);
%put &l;
aa bb cc
  */

   %ds2ma(in_ds=&in_ds,var=&var,uniq=&uniq,ma=_mz,where=%superq(where));
   %global &list;
   %let &list= %ma2l(ma=_mz);

%mend;


%macro ds2ma(in_ds=, var=, ma=, uniq=1, num=0, where=,format=);
 /* dataset variable values to macro array
  in_ds() - dataset name
  var - name of variable to make into macro array
  ma - name of macro variable prefix
  options
  uniq - 1 if to be sorted with uniq values, 0 if as is
  where - optional proc sql-type where clause for filtering in_ds
  format - optional format
  Example:
  data x;
  format a b $8.;
  input a $ b $;
  cards;
aa ba
ab bb
ac bc
;
%ds2ma(in_ds=x, var=a, ma=z);
%put &z0 &z1 &z2 &z3;
3 aa ab ac
  */
   %local i;
   %global &ma.0;

   %let &ma.0 = 0;

  %if &uniq eq 1 %then %do;
     proc sql noprint;
     select count(distinct &var)
        into :&ma.0
        from &in_ds
        where &var ne %if &num eq 0 %then ''; %else .;
     %if "%superq(where)" ne "" %then and %str(%superq(where));
     ;

     %if &&&ma.0 ne 0 %then %do;
        %let &ma.0 = &&&ma.0;

        %do i = 1 %to &&&ma.0;
           %global &ma&i;
        %end;

        proc sql noprint;
        select distinct  &var %if "&format" ne "" %then format &format ;
           into :&ma.1 - :&&ma&&&ma.0
           from &in_ds
           where &var ne %if &num eq 0 %then ''; %else .;
     %if "%superq(where)" ne "" %then and %superq(where);
        order by &var;

     %end;
  %end;
  %else %do;
     proc sql noprint;
     select count(*)
        into :&ma.0
        from &in_ds
     %if "%superq(where)" ne "" %then where %superq(where);
;

     %if &&&ma.0 ne 0 %then %do;
        %let &ma.0 = &&&ma.0;

        %do i = 1 %to &&&ma.0;
          %global &ma&i;
        %end;

        proc sql noprint;
        select &var %if "&format" ne "" %then format &format ;
           into :&ma.1 - :&&ma&&&ma.0
           from &in_ds
     %if "%superq(where)" ne "" %then where %superq(where);
;

     %end;

  %end;
%mend;

%macro dsv2ma(in_ds=, drop=, ma=,sort=1);
 /* dataset to  macro array: makes list out of variables in a dataset
  input
  in_ds() input dataset
  drop - list of variables to skip
  output
  ma - macro array prefix
  sort - use sort=0 if you want to preserve the order of the variables, otherwise they
'll be sorted by their name

  Example:
  data x;
```

```
   format a b $8.;
   input a $ b $;
   cards;
aa ba
ab bb
ac bc
;
%dsv2ma(in_ds=x, ma=z);
%put &z0 &z1 &z2;
2 A B
   */

    proc contents data=&in_ds(drop = &drop)
       out=_m1(keep=name varnum ) short noprint;
    run;
    %if &sort eq 0 %then %do;
       proc sort data=_m1;
       by varnum;
    %end;

    %ds2ma(in_ds=_m1, var=name, ma=&ma %if &sort eq 0 %then ,uniq=0;);
    %trash(_m1);

%mend;

%macro ds2macro(in_ds=);
/* one-record database to macro variables
each variable's value is assigned to a macro variable of the same name
Example:
 data x;
 input a $ b $;
cards;
try this
;
%ds2macro(in_ds=x);
%put &a &b;
try this
*/
    %dsv2ma(in_ds=&in_ds,ma=_dm);
    %local i vars varscl x;
    %do i = 1 %to &_dm0;
       %global &&_dm&i;
    %end;

    %let varscl = %ma2l(ma=_dm, sep=%quote(,));
    %let varsccl = %ma2l(ma=_dm, sep=%quote(, ),prefix=%quote(:));
    proc sql noprint;
    select &varscl into &varsccl
       from &in_ds;

    %do i = 1 %to &_dm0;
       %let x = &&_dm&i;
       %let &&_dm&i = &&&x;
    %end;

%mend;


%macro dsv2FL(in_ds=,drop=,mf=,ml=,cnt=);
/* get first, last variable from a dataset, excluding &drop variables
in_ds() input dataset
drop - list of variables to ignore: these variables must be either first or last in th
e dataset
mf - name of macro to put first variable name into
ml - name of macro to put last  variable name into
cnt- name of macro to put number of variables into

  Example:
  data x;
  format a b c d $8.;
  input a $ b $ c $ d $;
  cards;
aa ba ca da
ab bb cb db
ac bc cc dc
;
%dsv2FL(in_ds=x, mf=from, ml=to, cnt=count);
%put &from &to &count;
A        D        4
*/
    %global &mf &ml &cnt;
    %let drop = %upcase(&drop);

    proc contents data=&in_ds(drop = &drop)
       out=_m1(keep=name varnum) short noprint;
    run;
    proc sort data=_m1;
    by varnum;
    run;
    data _null_;
    set _m1 end=eof;
    if _n_ = 1 then call symput("&mf",name);
    if eof then do;
       call symput("&ml",name);
       %if &cnt ne %then %do; call symput("&cnt",trim(left(put(_n_,8.)))); %end;
    end;
    run;
    %trash(_m1);

%mend;

%macro dsv2ds(in_ds=, drop=, var=, out_ds=);
 /* dataset variables to dataset of variable names
  input
  in_ds() input dataset
  drop - list of variables to skip
  var - name of variable in out_ds
  output
  out_ds(&var)

  Example:
  data x;
  format a b c d $8.;
  input a $ b $ c $ d $;
  cards;
aa ba ca da
ab bb cb db
ac bc cc dc
;
%dsv2ds(in_ds=x, drop=c, var=xv, out_ds=z);
  proc print data=z;run;
OBS XV
--- --
1   A
2   B
3   D
  */

    proc contents data=&in_ds(drop = &drop)
       out=&out_ds(keep=varnum name rename=(name=&var)) short noprint;
```

```
   run;
   proc sort data=&out_ds out=&out_ds(drop=varnum);
   by varnum;
   run;

%mend;


%macro l2ds(list=, var=, format=$8., out_ds=);
 /* list to dataset
  input
  list - blank-separated list
  var - name of variable in &out_ds
  format - format of variable in &out_ds
  output
  out_ds(&var) - dataset with one obs per item on list

  Example
%let l = FMAGX FMILX FCONX
%l2ds(list= FMAGX FMILX FCONX, var=xv, out_ds=z);
  proc print data=z;run;
OBS XV
--- --
1   FMAGX
2   FMILX
3   FCONX
  */
   %local i;

   data &out_ds;
   format &var &format;
   %let i = 1;
   %let _l = %scan(&list,&i,%str( ));
   %do %while ("&_l" ne "");
      %if %substr(&format,1,1) eq $ %then
         &var = "&_l";
      %else
         &var = &_l;;
      output;
      %let i = %eval(&i + 1);
      %let _l = %scan(&list,&i,%str( ));
   %end;
   run;

%mend;

%macro l2ma(list=, ma=);
 /* list to macro array
  input
  list - macro list
  output
  ma - prefix of macro array

  Example
%l2ma(list=FMAGX FMILX FCONX,ma=t);
%put &t0 &t1 &t2 &t3;
3 FMAGX FMILX FCONX
  */

   %global &ma.0;
   %global &ma.1;

   %local i;
   %let i = 1;
   %let &ma.1 = %scan( &list, &i, %str( ) );

   %do %while ("&&&ma&i"  ne "");

      %let i = %eval(&i + 1);
      %global &ma&i;
      %let &ma&i = %scan( &list, &i, %str( ) );

   %end;

   %let &ma.0 = %eval(&i - 1);

%mend ;


%macro ma2l(ma=, sep=%quote( ), prefix=, suffix=);
/* macro array to list
 sep can be a comma if inside a %qoute()
 prefix and suffix can be used to put quotes or whatever around values

Example 1:
%put &t0 &t1 &t2 &t3;
3 FMAGX FMILX FCONX
%let l = %ma2l(ma=t);
%put &l;
FMAGX FMILX FCONX

Example 2:
%put &t0 &t1 &t2 &t3;
3 FMAGX FMILX FCONX
%let l = %ma2l(ma=t, sep=%quote(, ));
%put &l;
FMAGX, FMILX, FCONX
*/

   %local i;
   %if &&&ma.0 ne 0 %then &prefix.&&&ma.1&suffix;%do i = 2 %to &&&ma.0;&sep.&prefix.&&
&ma&i.&suffix %end;

%mend;

%macro ma2qcl(ma=);
 /* macro array to quoted, comma-separated list

Example:
%put &t0 &t1 &t2 &t3;
3 FMAGX FMILX FCONX
%let l = %ma2qcl(ma=t);
"FMAGX","FMILX","FCONX"
*/
   %local i ol;
   %let ol = %str(%')&&&ma.1%str(%');
   %do i = 2 %to &&&ma.0;
         %let ol = &ol,"&&&ma&i%";
   %end;
   %str(&ol)
%mend;

%macro l2qcl(list);
 /* list to quoted, comma-separated list

Example:
%let l = FCONX FMAGX FMILX CNGRX;
%let qcl = %l2qcl(&l);
%put &qcl;
"FCONX","FMAGX","FMILX","CNGRX"
```

```
*/
   %local i l ol;
   %let i = 1;
   %let ol = ;
   %let l = %trim(%scan( &list,&i, %str( ) ));
   %do %while (&l ne );
      %if &ol ne %then %let ol = &ol,;
      %let ol = &ol."&l";
      %let i = %eval(&i + 1);
      %let l = %trim(%scan( &list,&i, %str( ) ));
   %end;
   %str(&ol)
%mend;


%macro l2cl(list);
 /* list to comma-separated list

Example:
%let l = FCONX FMAGX FMILX CNGRX;
%let cl = %l2cl(&l);
%put &cl;
FCONX,FMAGX,FMILX,CNGRX
*/
   %local i l ol;
   %let i = 1;
   %let ol = ;
   %let l = %trim(%scan( &list,&i, %str( ) ));
   %do %while (&l ne );
      %if &ol ne %then %let ol = &ol,;
      %let ol = &ol.&l;
      %let i = %eval(&i + 1);
      %let l = %trim(%scan( &list,&i, %str( ) ));
   %end;
   %str(&ol)
%mend;

%macro ma2ds(ma=, var=, format=$8., out_ds=ds);
 /* macro array to dataset
  input
  ma - a macro array prefix
  varname - name of var in ds
  output
  ds(&var)

Example
%put &t0 &t1 &t2 &t3;
3 FMAGX FMILX FCONX
%ma2ds(ma=t, var=ticker, out_ds=z);
proc print data=z;run;
OBS TICKER
1   FMAGX
2   FMILX
3   FCONX
  */
   data &out_ds;
   format &var &format;
   %local i;
   %do i = 1 %to &&&ma.0;
      %if %substr(&format,1,1) eq $ %then
         &var = "&&&ma&i";
      %else
         &var = &&&ma&i;;
      output;
   %end;
   run;
%mend;


%macro mama2ma(maa=, mab=, mac=);
 /* merge 2 macro arrays into a 3rd one
  input
  maa - a macro array prefix
  mab - a macro array prefix
  output
  mac - output macro array prefix

Example
%put &s0 &s1 &s2;
2 CNGRX FLPSX
%put &t0 &t1 &t2 &t3;
3 FMAGX FMILX FCONX
%mama2ma(maa=t, mab=s, mac=u);
%put &u0 &u1 &u2 &u3 &u4 &u5;
5 FMAGX FMILX FCONX CNGRX FLPSX
  */
   %local i j;
   %global &mac.0;

   data _m1;
   format var $200.;
   %do i = 1 %to &&&maa.0;
      %global &mac&i;
      %let &mac.0 = &i;
      %let &mac&i = &&&maa&i;
      var = "&&&maa&i";
      output;
   %end;

   %do i = 1 %to &&&mab.0;
      %let j = %eval(&i + &&&maa.0);
      %global &mac&j;
      %let &mac&j = &&&mab&i;
      var = "&&&mab&i";
      output;
   %end;
   run;
   %let &mac.0 = %eval(&&&maa.0 + &&&mab.0);
   %trash(_m1);
%mend;

%macro ma_ma(maa=, mab=, mac=, op=and not);
 /* boolean operation on  2 macro arrays
  input
  maa - a macro array prefix
  mab - a macro array prefix
  output
  mac - a macro array prefix.
  options
  op - operation to perform.  defaults to "and not", which is a subtraction

Example 1:
%put &s0 &s1 &s2;
3 CNGRX FLPSX FBGRX
%put &u0 &u1 &u2 &u3 &u4 &u5;
5 FMAGX FMILX FCONX CNGRX FLPSX
%ma_ma(maa=u,mab=s,mac=t);
%put &t0 &t1 &t2 &t3 ;
3 FCONX FMAGX FMILX
```

```
Example 2
%ma_ma(maa=u,mab=s,mac=v,op=OR);
%put &v0 &v1 &v2 &v3 &v4 &v5 &v6;
6 CNGRX FBGRX FCONX FLPSX FMAGX FMILX
  */
   %local i j;
   %ma2ds(ma=&maa,var=x,out_ds=_m2);
   %ma2ds(ma=&mab,var=x,out_ds=_m3);
   proc sort data=_m2; by x;
   proc sort data=_m3; by x;
   data _m4;
   merge _m2(in=_ina) _m3(in=_inb);
   by x;
   if _ina &op _inb;
   run;
   %ds2ma(in_ds=_m4,var=x,ma=&mac);
   %trash(_m2 _m3 _m4);
%mend;

%macro ma_app(ma=,value=);
/* append an item to a macro array
 ma - macro array previx
 value - value to be added to end of array

Example:
%put &t0 &t1 &t2 &t3 ;
3 FCONX FMAGX FMILX
%ma_app(ma=t,value=CNGRX);
 %put &t0 &t1 &t2 &t3 &t4 ;
4 FCONX FMAGX FMILX CNGRX
*/

   %local i;
   %let i = %eval(&&&ma.0+1);
   %let &ma.0 = &i;
   %global &&ma&&&ma.0;
   %let &&ma&&&ma.0 = &value;

%mend;


%macro invertma(ma=);
/* invert a macro array
create macro variables for each item i in ma X whose names
are the values of Xi and whose values are i

Example:
 %put &t0 &t1 &t2 &t3 &t4 ;
4 FCONX FMAGX FMILX CNGRX
%invertma(ma=t);
%put &FCONX &FMAGX &FMILX &CNGRX;
1 2 3 4
*/

   %local _i;
   %do _i = 1 %to &&&ma.0;
      %global &&&ma&_i;
      %let &&&ma&_i = &_i;
   %end;

%mend;

%macro lib2ma(lib=, ma=);
/* names of datasets in library to macro array
lib - library name
ma - name of macro array

Example:
 x 'ls -1 /home/bvick/foo';
/home/bvick/foo/foo.ssd01
/home/bvick/foo/gs.ssd01
/home/bvick/foo/pb1.ssd01
/home/bvick/foo/ray_ret.ssd01
 libname ll  '/home/bvick/foo';
 %lib2ma(lib=ll,ma=f);
%put &f0 &f1 &f2 &f3 &f4;
4 FOO GS PB1 RAY_RET
*/

   proc contents data=&lib.._all_  memtype=data noprint out=_l2ma;
   run;

   data _l2ma;
   set _l2ma(keep=memname);
   by memname;
   if first.memname;
   run;

   %ds2ma(in_ds=_l2ma, var=memname, ma=&ma);
   %trash(_l2ma);

%mend;
```

macros.macro


```
%include "&FPPCODE/utilities/date.macro";
%include "&FPPCODE/utilities/keyvar.macro";
%include "&FPPCODE/utilities/assets.macro";
%include "&FPPCODE/utilities/macroman.macro";
%include "&FPPCODE/utilities/dsutil.macro";
%include "&FPPCODE/utilities/graph.macro";
%include "&FPPCODE/pop/optimizer/compositer.macro";
%include "&FPPCODE/pop/optimizer/quad.macro";
%include "&FPPCODE/pop/optimizer/linear_regressor.macro";
%include "&FPPCODE/pop/optimizer/retcalc.macro";
%include "&FPPCODE/ref/refdata.macro";
%include "&FPPCODE/fmr_equity/anal.macro";
```

```
/* macros to get reference data
  To call these, first:

 %let dsbase  = %sysget(DSBASE);  (normally /appdata/watermdl)
 %let fppcode = %sysget(FPPCODE); (normally /project/quantitative1/fppadmin/projects)

 %include "&fppcode/utilities/macros.macro";


*/

%macro d_stkret(from_dt=, to_dt=, in_tv=, tickers=, out_dtr=, out_dtt=, missing=,frn=1
,dom=1);
/* get daily stock returns for date range
from_dt yyyymmdd from date
to_dt   yyyymmdd from date
choose tickers | in_tv | neither
tickers - optional list of tickers to get returns for
in_tv(ticker,...) optional dataset of tickers to get returns for
out_dtr(date,ticker,return) output return series
out_dtt(date,ticker,ticker...) output return series
missing(ticker,count) optional list of tickers for which there aren't a full set of re
turns - requires &tickers or &in_tv
frn - nonzero indicates include foreign stocks
dom - nonzero indicates include domestic stocks
*/



   %if &from_dt eq %then %let from_dt=0;
   %if &to_dt eq %then %let to_dt = 99999999;
   %if &out_dtr eq %then %let out_dtr=_outdtr;

   %local d fy ty got rds;
   %let fy = %eval(&from_dt/10000);
   %let ty = %eval(&to_dt  /10000);
   %let got = 1;
   %let rds = &out_dtr;
   %if &in_tv ne %then %do;
      proc sql;
      create table _tick as
         select distinct ticker from &in_tv order by ticker;
   %end;
   %else %if "&tickers" ne "" %then %do;
      %l2ds(list=&tickers,out_ds=_tick,var=ticker);
      %let in_tv=1;
      proc sort data=_tick;
      by ticker;
      run;
   %end;
   %do d = &fy %to &ty;
      %if %sysfunc(exist(ref.dstk&d)) and &dom ne 0 %then %do;

         proc sql noprint;
         create table &rds as
            select date,a.ticker,return
            from ref.dstk&d a %if &in_tv ne %then , _tick b;
         where a.date >= &from_dt and a.date <=&to_dt
            %if &in_tv ne %then and a.ticker=b.ticker;
         order by date;
         %if &got ne 1 %then %do;
            proc append base=&out_dtr data= _sr;
            run;
         %end;
         %let got = 2;
         %let rds =  _sr;
      %end;
      %if %sysfunc(exist(ref.dfrn&d)) and &frn ne 0 %then %do;
         proc sql noprint;
         create table &rds as
            select date,a.ticker,return
            from ref.dfrn&d a %if &in_tv ne %then , _tick b;
         where a.date >= &from_dt and a.date <=&to_dt
            %if &in_tv ne %then and a.ticker=b.ticker;
         order by date;
         %if &got ne 1 %then %do;
            proc append base=&out_dtr data= _sr;
            run;
         %end;
         %let got = 2;
         %let rds =  _sr;
      %end;
   %end;
   proc sort data=&out_dtr;
   by date ticker;
   run;

/* remove duplicates */
   data &out_dtr;
   set &out_dtr;
   by date ticker;
   if first.ticker;
   run;


   %if &missing ne %then %do;
      %local _maxcnt;

      %holiday0(in_dtr=&out_dtr,out_dtr=&out_dtr,ref_dtt=ref._hol);

      proc sql noprint;
      create table _cnt as
         select ticker,count(*) as count
         from &out_dtr
         group by ticker
         order by ticker;
      select max(count)
         into :_maxcnt
         from _cnt;
      data &missing(keep=ticker count);
      merge _tick(in=_int) _cnt(in=_inc);
      by ticker;
      if _int;
      if not _inc then count=0;
      if count ne &_maxcnt then output;
      run;
      %trash(_tick _cnt);
   %end;

   %if &out_dtt ne %then %do;
      %dtr2dtt(in_dtr=&out_dtr,out_dtt=&out_dtt);
      %trash(_outdtr);
   %end;

%mend;


%macro d_idxret(from_dt=, to_dt=, in_tv=, tickers=, out_dtr=, out_dtt=);
/* get daily index returns for date range
from_dt yyyymmdd from date
```

```
to_dt   yyyymmdd from date
choose tickers | in_tv | neither
tickers - optional list of tickers to get returns for
in_tv(ticker,...) optional dataset of tickers to get returns for
out_dtr(date,ticker,return) output return series
out_dtt(date,ticker,ticker...) output return series
*/

   %if &from_dt eq %then %let from_dt=0;
   %if &to_dt eq %then %let to_dt = 99999999;
   %if &out_dtr eq %then %let out_dtr=_outdtr;
   %if &in_tv ne %then %do;
      proc sql;
      create table _tick as
         select distinct ticker from &in_tv order by ticker;
   %end;
   %else %if "&tickers" ne "" %then %do;
      %l2ds(list=&tickers,out_ds=_tick,var=ticker);
      %let in_tv=1;
      proc sort data=_tick;
      by ticker;
      run;
   %end;
   proc sql;
   create table &out_dtr as
      select a.date,a.ticker,a.return
      from ref.id_dtr a %if &in_tv ne %then %do; , _tick b %end;
   where a.date >= &from_dt and a.date <= &to_dt
      %if &in_tv ne %then %do; and a.ticker=b.ticker %end;
   order by date,ticker;

   %if &out_dtt ne %then %do;
      %dtr2dtt(in_dtr=&out_dtr,out_dtt=&out_dtt);
      %trash(_outdtr);
   %end;
   %if &in_tv ne %then %trash(_tick);

%mend;


%macro d_fndret(from_dt=, to_dt=, in_tv=, tickers=, out_dtr=, out_dtt=, missing=, mias
=0);
/* get daily fund returns for date range
from_dt yyyymmdd from date
to_dt   yyyymmdd from date
choose tickers | in_tv | neither
tickers - optional list of tickers to get returns for
in_tv(ticker,...) optional dataset of tickers to get returns for
out_dtr(date,ticker,return) output return series
out_dtt(date,ticker,ticker...) output return series
mias - nonzero indicate also look into mias returns
*/



   %if &from_dt eq %then %let from_dt=0;
   %if &to_dt eq %then %let to_dt = 99999999;
   %if &out_dtr eq %then %let out_dtr=_outdtr;

   %local d fy ty got rds;
   %let fy = %eval(&from_dt/10000);
   %let ty = %eval(&to_dt  /10000);
   %let got = 1;
   %let rds = &out_dtr;
   %if &in_tv ne %then %do;
      proc sql;
      create table _tick as
        select distinct ticker from &in_tv order by ticker;
   %end;
   %else %if "&tickers" ne "" %then %do;
      %l2ds(list=&tickers,out_ds=_tick,var=ticker);
      %let in_tv=1;
      proc sort data=_tick;
      by ticker;
      run;
   %end;
   %do d = &fy %to &ty;
      %if %sysfunc(exist(ref.fdr&d)) %then %do;

         proc sql noprint;
         create table &rds as
            select date,a.ticker,return
            from ref.fdr&d a %if &in_tv ne %then , _tick b;
         where a.date >= &from_dt and a.date <=&to_dt
            %if &in_tv ne %then and a.ticker=b.ticker;
         order by date;
         %if &got ne 1 %then %do;
            proc append base=&out_dtr data= _sr;
            run;
         %end;
         %let got = 2;
         %let rds =  _sr;
      %end;
   %end;
   %if &mias ne 0 %then %do;
      proc sql noprint;
      create table &rds as
         select date,a.ticker,return
         from ref.fmd_dtr a %if &in_tv ne %then , _tick b;
         where a.date >= &from_dt and a.date <=&to_dt
            %if &in_tv ne %then and a.ticker=b.ticker;
         order by date;
         %if &got ne 1 %then %do;
           proc append base=&out_dtr data= _sr;
           run;
         %end;
         %let got = 2;
         %let rds =  _sr;
   %end;

   proc sort data=&out_dtr;
   by date ticker;
   run;

/* remove duplicates */
   data &out_dtr;
   set &out_dtr;
   by date ticker;
   if first.ticker;
   run;


   %if &missing ne %then %do;
      %local _maxcnt;

      %holiday0(in_dtr=&out_dtr,out_dtr=&out_dtr,ref_dtt=ref._hol);

      proc sql noprint;
      create table _cnt as
         select ticker,count(*) as count
```

```
         from &out_dtr
         group by ticker
         order by ticker;
      select max(count)
         into :_maxcnt
         from _cnt;
      data &missing(keep=ticker count);
      merge _tick(in=_int) _cnt(in=_inc);
      by ticker;
      if _int;
      if not _inc then count=0;
      if count ne &_maxcnt then output;
      run;
      %trash(_tick _cnt);
   %end;

   %if &out_dtt ne %then %do;
      %dtr2dtt(in_dtr=&out_dtr,out_dtt=&out_dtt);
      %trash(_outdtr);
   %end;


%mend;

%macro d_ret(from_dt=, to_dt=, in_tv=, tickers=, out_dtr=, out_dtt=);
/* get daily fund,index and/or stock returns
from_dt yyyymmdd from date
to_dt   yyyymmdd from date
choose tickers | in_tv | neither
tickers - optional list of tickers to get returns for
in_tv(ticker,...) optional dataset of tickers to get returns for
out_dtr(date,ticker,return) output return series
out_dtt(date,ticker,ticker...) output return series
this just calls the other macros, so if you have all one type, use the specific ones
*/

   %d_stkret(from_dt=&from_dt, to_dt=&to_dt, in_tv=&in_tv, tickers=&tickers,
             out_dtr=%if &out_dtr ne %then _dr1;,
             out_dtt=%if &out_dtt ne %then _dt1;);
   %d_idxret(from_dt=&from_dt, to_dt=&to_dt, in_tv=&in_tv, tickers=&tickers,
             out_dtr=%if &out_dtr ne %then _dr2;,
             out_dtt=%if &out_dtt ne %then _dt2;);
   %d_fndret(from_dt=&from_dt, to_dt=&to_dt, in_tv=&in_tv, tickers=&tickers,
             out_dtr=%if &out_dtr ne %then _dr3;,
             out_dtt=%if &out_dtt ne %then _dt3;);
   %if &out_dtr ne %then %do;
      data &out_dtr;
      set _dr1 _dr2 _dr3;
      run;
      proc sort data=&out_dtr;
      by date ticker;
      run;
   %end;
   %if &out_dtt ne %then %do;
      data &out_dtt;
      merge _dt1 _dt2 _dt3;
      by date;
      run;
   %end;

%mend;



%macro m_stkret(from_dt=, to_dt=, in_tv=, tickers=, out_dtr=, out_dtt=);
/* get monthly returns of stocks for date range
from_dt yyyymm from date
to_dt   yyyymm from date
choose tickers | in_tv | neither
tickers - optional list of tickers to get returns for
in_tv(ticker,...) optional dataset of tickers to get returns for
out_dtr(date,ticker,return) output return series
out_dtt(date,ticker,ticker...) output return series
dates are in yyyymm form
*/

   %if &out_dtr eq %then %let out_dtr=_outdtr;
   %if &in_tv ne %then %do;
      proc sql;
      create table _tick as
         select distinct ticker from &in_tv order by ticker;
   %end;
   %else %if "&tickers" ne "" %then %do;
      %l2ds(list=&tickers,out_ds=_tick,var=ticker);
      %let in_tv=1;
      proc sort data=_tick;
      by ticker;
      run;
   %end;
   %if &from_dt eq %then %let from_dt = 0;
   %if &to_dt eq %then %let to_dt = 999999;


   proc sql;
   create table &out_dtr as
      select date,a.ticker,a.return
      from ref.sm_dtr a %if &in_tv ne %then %do; , _tick b %end;
      where a.date >= &from_dt and a.date <= &to_dt
      %if &in_tv ne %then %do; and a.ticker=b.ticker %end;
   order by date,ticker;

   %if &out_dtt ne %then %do;
      %dtr2dtt(in_dtr=&out_dtr,out_dtt=&out_dtt);
      %trash(_outdtr);
   %end;

   %if &in_tv ne %then %trash(_tick);

%mend;

%macro m_idxret(from_dt=, to_dt=, in_tv=, tickers=, out_dtr=, out_dtt=);
/* get monthly returns of indexes for date range
from_dt yyyymm from date
to_dt   yyyymm from date
choose tickers | in_tv | neither
tickers - optional list of tickers to get returns for
in_tv(ticker,...) optional dataset of tickers to get returns for
out_dtr(date,ticker,return) output return series
out_dtt(date,ticker,ticker...) output return series
dates are in yyyymm form
*/
   %if &out_dtr eq %then %let out_dtr=_outdtr;
   %if &in_tv ne %then %do;
      proc sql;
      create table _tick as
         select distinct ticker from &in_tv order by ticker;
   %end;
   %else %if "&tickers" ne "" %then %do;
      %l2ds(list=&tickers,out_ds=_tick,var=ticker);
      %let in_tv=1;
```

```
      proc sort data=_tick;
      by ticker;
      run;
   %end;
   %if &from_dt eq %then %let from_dt = 0;
   %if &to_dt eq %then %let to_dt = 999999;


   proc sql;
   create table &out_dtr as
      select date,a.ticker,a.return
      from ref.im_dtr a %if &in_tv ne %then %do; , _tick b %end;
      where a.date >= &from_dt and a.date <= &to_dt
      %if &in_tv ne %then %do; and a.ticker=b.ticker %end;
   order by date,ticker;

   %if &out_dtt ne %then %do;
      %dtr2dtt(in_dtr=&out_dtr,out_dtt=&out_dtt);
      %trash(_outdtr);
   %end;

   %if &in_tv ne %then %trash(_tick);

%mend;

%macro m_fndret(from_dt=, to_dt=, in_tv=, tickers=, out_dtr=, out_dtt=);
/* get monthly returns of funds for date range
from_dt yyyymm from date
to_dt   yyyymm from date
choose tickers | in_tv | neither
tickers - optional list of tickers to get returns for
in_tv(ticker,...) optional dataset of tickers to get returns for
out_dtr(date,ticker,return) output return series
out_dtt(date,ticker,ticker...) output return series
dates are in yyyymm form
*/
   %if &out_dtr eq %then %let out_dtr=_outdtr;
   %if &in_tv ne %then %do;
      proc sql;
      create table _tick as
         select distinct ticker from &in_tv order by ticker;
   %end;
   %else %if "&tickers" ne "" %then %do;
      %l2ds(list=&tickers,out_ds=_tick,var=ticker);
      %let in_tv=1;
      proc sort data=_tick;
      by ticker;
      run;
   %end;
   %if &from_dt eq %then %let from_dt = 0;
   %if &to_dt eq %then %let to_dt = 999999;


   proc sql;
   create table &out_dtr as
      select date,a.ticker,a.return
      from ref.fm_dtr a %if &in_tv ne %then %do; , _tick b %end;
      where a.date >= &from_dt and a.date <= &to_dt
      %if &in_tv ne %then %do; and a.ticker=b.ticker %end;
   order by date,ticker;

   %if &out_dtt ne %then %do;
      %dtr2dtt(in_dtr=&out_dtr,out_dtt=&out_dtt);
      %trash(_outdtr);
   %end;

   %if &in_tv ne %then %trash(_tick);

%mend;

%macro get_returns(from_dt=, to_dt=, in_tv=, tickers=, out_dtr=, out_dtt=);
/* get daily/monthly fund,index and/or stock returns
from_dt yyyymm or yyyymmdd from date
to_dt   yyyymm or yyyymmdd to  date
choose tickers | in_tv | neither
tickers - optional list of tickers to get returns for
in_tv(ticker,...) optional dataset of tickers to get returns for
out_dtr(date,ticker,return) output return series
out_dtt(date,ticker,ticker...) output return series
this just calls the other macros, so if you have all one type, use the specific ones
*/

   %if %eval(%length(&from_dt) eq 6) %then %do;


      %m_stkret(from_dt=&from_dt, to_dt=&to_dt, in_tv=&in_tv, tickers=&tickers,
                out_dtr=%if &out_dtr ne %then _dr1;,
                out_dtt=%if &out_dtt ne %then _dt1;);
      %m_idxret(from_dt=&from_dt, to_dt=&to_dt, in_tv=&in_tv, tickers=&tickers,
                out_dtr=%if &out_dtr ne %then _dr2;,
                out_dtt=%if &out_dtt ne %then _dt2;);
      %m_fndret(from_dt=&from_dt, to_dt=&to_dt, in_tv=&in_tv, tickers=&tickers,
                out_dtr=%if &out_dtr ne %then _dr3;,
                out_dtt=%if &out_dtt ne %then _dt3;);
   %end;
   %else %do;
      %d_stkret(from_dt=&from_dt, to_dt=&to_dt, in_tv=&in_tv, tickers=&tickers,
                out_dtr=%if &out_dtr ne %then _dr1;,
                out_dtt=%if &out_dtt ne %then _dt1;);
      %d_idxret(from_dt=&from_dt, to_dt=&to_dt, in_tv=&in_tv, tickers=&tickers,
                out_dtr=%if &out_dtr ne %then _dr2;,
                out_dtt=%if &out_dtt ne %then _dt2;);
      %d_fndret(from_dt=&from_dt, to_dt=&to_dt, in_tv=&in_tv, tickers=&tickers,
                out_dtr=%if &out_dtr ne %then _dr3;,
                out_dtt=%if &out_dtt ne %then _dt3;);
   %end;

   %if &out_dtr ne %then %do;
      data &out_dtr;
      set _dr1 _dr2 _dr3;
      run;
      proc sort data=&out_dtr;
      by date ticker;
      run;
   %end;
   %if &out_dtt ne %then %do;
      data &out_dtt;
      merge _dt1 _dt2 _dt3;
      by date;
      run;
   %end;

%mend;



%macro r_stkchr(in_tv=, tickers=, out_tv=);
/* get stock characteristics
choose tickers | in_tv | neither
tickers - optional list of tickers to get returns for
```

```
in_tv(ticker,...) optional dataset of tickers to get returns for
out_tv(ticker,...) output dataset of stock characteristics
*/
   %if &in_tv ne %then %do;
      proc sql;
      create table _tick as
         select distinct ticker from &in_tv order by ticker;
   %end;
   %else %if "&tickers" ne "" %then %do;
      %l2ds(list=&tickers,out_ds=_tick,var=ticker);
      %let in_tv=1;
      proc sort data=_tick;
      by ticker;
      run;
   %end;

   %if &in_tv ne %then %do;
      data &out_tv;
      merge ref.fame_eq _tick(in=_ina);
      by ticker;
      if _ina;
      run;
      %trash(_tick);
   %end;
   %else %do;
      data &out_tv;
      set ref.fame_eq;
      run;
   %end;


%mend;


%macro r_fndchr(in_tv=, tickers=, out_tv=);
/* get fund characteristics
choose tickers | in_tv | neither
tickers - optional list of tickers to get returns for
in_tv(ticker,...) optional dataset of tickers to get returns for
out_tv(ticker,...) output dataset of fund characteristics
*/
   %if &in_tv ne %then %do;
      proc sql;
      create table _tick as
         select distinct ticker from &in_tv order by ticker;
   %end;
   %else %if "&tickers" ne "" %then %do;
      %l2ds(list=&tickers,out_ds=_tick,var=ticker);
      %let in_tv=1;
      proc sort data=_tick;
      by ticker;
      run;
   %end;

   %if &in_tv ne %then %do;
      data &out_tv;
      merge ref.ms_mast _tick(in=_ina);
      by ticker;
      if _ina;
      run;
      %trash(_tick);
   %end;
   %else %do;
      data &out_tv;
      set ref.ms_mast;
      run;
   %end;

%mend;


%macro m_fndcon(from_dt=, to_dt=, in_tv=,tickers=, out_ctw=, out_dctw=);
/* fund constituent holdings
from_dt yyyymm from date
to_dt   yyyymm from date
choose tickers | in_tv | neither
tickers - optional list of tickers to get holdings for
in_tv(ticker,...) optional dataset of tickers to get holdings for
choose out_ctw if a single date &from_dt | out_dctw
out_ctw(cticker,ticker,wgt)
out_dctw(date,cticker,ticker,wgt)
dates are in yyyymm form
*/
   %if &in_tv ne %then %do;
      proc sql;
      create table _tick as
         select distinct ticker from &in_tv order by ticker;
   %end;
   %else %if "&tickers" ne "" %then %do;
      %l2ds(list=&tickers,out_ds=_tick,var=ticker);
      %let in_tv=1;
      proc sort data=_tick;
      by ticker;
      run;
   %end;
   %if &from_dt eq %then %let from_dt = 0;
   %if &to_dt eq %then %let to_dt = 999999;



   %if &out_dctw ne %then %do;
      proc sql;
      create table &out_dctw as
         select date,a.cticker,a.ticker,a.wgt
         from ref.m_fndcon a %if &in_tv ne %then %do; , _tick b %end;
         where a.date >= &from_dt and a.date <= &to_dt
         %if &in_tv ne %then %do; and a.cticker=b.ticker %end;
         order by date,cticker,ticker;

   %end;
   %if &out_ctw ne %then %do;
      proc sql;
      create table &out_ctw as
         select a.cticker,a.ticker,a.wgt
         from ref.m_fndcon a %if &in_tv ne %then %do; , _tick b %end;
         where a.date = &from_dt
         %if &in_tv ne %then %do; and a.cticker=b.ticker %end;
         order by cticker,ticker;

   %end;

   %if &in_tv ne %then %trash(_tick);

%mend;


%macro m_idxcon(from_dt=, to_dt=, in_tv=, tickers=, out_ctw=, out_dctw=);
/* index constituents
from_dt yyyymm from date
to_dt   yyyymm from date
```

```
choose tickers | in_tv | neither
tickers - optional list of tickers to get constituents for
in_tv(ticker,...) optional dataset of tickers to get constituents for
choose (out_ctw if a single date &from_dt | out_dctw)
out_ctw(cticker,ticker,wgt)
out_dctw(date,cticker,ticker,wgt)
dates are in yyyymm form
*/
   %if &in_tv ne %then %do;
      proc sql;
      create table _tick as
         select distinct ticker from &in_tv order by ticker;
   %end;
   %else %if "&tickers" ne "" %then %do;
      %l2ds(list=&tickers,out_ds=_tick,var=ticker);
      %let in_tv=1;
      proc sort data=_tick;
      by ticker;
      run;
   %end;
   %if &from_dt eq %then %let from_dt = 0;
   %if &to_dt eq %then %let to_dt = 999999;

   %if &out_dctw ne %then %do;
      proc sql;
      create table &out_dctw as
         select date,a.cticker,a.ticker,a.wgt
         from ref.m_idxcon a %if &in_tv ne %then %do; , _tick b %end;
         where a.date >= &from_dt and a.date <= &to_dt
         %if &in_tv ne %then %do; and a.cticker=b.ticker %end;
         order by date,cticker,ticker;

   %end;
   %if &out_ctw ne %then %do;
      proc sql;
      create table &out_ctw as
         select a.cticker,a.ticker,a.wgt
         from ref.m_idxcon a %if &in_tv ne %then %do; , _tick b %end;
         where a.date = &from_dt
         %if &in_tv ne %then %do; and a.cticker=b.ticker %end;
         order by cticker,ticker;

   %end;

   %if &in_tv ne %then %trash(_tick);

%mend;


%macro r_stkcls(in_tv=, tickers=, scheme=, out_ctw=, out_tv=, var=);
/* stock classifications
choose tickers | in_tv | neither
tickers - optional list of tickers to get constituents for
in_tv(ticker,...) optional dataset of tickers to get constituents for
scheme - one of (MSECT,FMRIND,GICS,GICI,GICSUB,LMMSECT,LMFMRIND,LMGICS,LMGICI, LMGICSU
B)
out_ctw(cticker,ticker,wgt) cticker is stock, ticker is sector id, wgt is 1
out_tv(ticker,&var,...) output stock-indu classification
var - name of variable in &out_tv to put sector classication into
Actually, classification can be done by any $8. or less variable in ref.stkchar, for e
xample CTRY.
*/
   %if &in_tv ne %then %do;
      proc sql;
      create table _tick as
         select distinct ticker from &in_tv order by ticker;
   %end;
   %else %if "&tickers" ne "" %then %do;
      %l2ds(list=&tickers,out_ds=_tick,var=ticker);
      %let in_tv=1;
      proc sort data=_tick;
      by ticker;
      run;
   %end;

   %if &out_ctw eq %then %do;
      %let out_ctw= _out_ctw;
   %end;

   proc sql noprint;
   create table &out_ctw as
      select a.ticker as cticker, b.&scheme as ticker, 1 as wgt
      from _tick a, ref.stkchar b
      where b.ticker ne ''
      and a.ticker eq b.ticker
      order by cticker;

   %if &out_tv ne %then %do;

      data &out_tv;
      set &out_ctw(rename=( ticker=&var cticker=ticker) drop=wgt);
      run;

      %if &out_ctw eq _out_ctw %then %trash(_out_ctw);;

   %end;


%mend;

libname ref "&dsbase/ref";
```

Obs	_TYPE_	_NAME_	CPO	FBNDX	FCNTX	FGRIX	FLPSX	FMAGX	FOSFX	FUSEX	_GIC_	_RHS_
1	QUAD	CPO	95.4425	3.5233	2.25706	2.86878	7.28982	-2.52313	1.52767	-1.37242	3.4492	.
2	QUAD	FBNDX	3.5233	20.6003	2.52222	4.03311	6.20536	-3.33476	-2.65059	-2.30571	18.7299	.
3	QUAD	FCNTX	2.2571	2.5222	4.64043	0.81344	0.64328	-0.67851	0.55160	-1.27919	2.3910	.
4	QUAD	FGRIX	2.8688	4.0331	0.81344	4.45965	0.69113	0.32769	-2.37666	1.13074	4.0689	.
5	QUAD	FLPSX	7.2898	6.2054	0.64328	0.69113	8.46446	-1.73866	-1.80041	-1.17474	6.1420	.
6	QUAD	FMAGX	-2.5231	-3.3348	-0.67851	0.32769	-1.73866	2.84754	-0.57389	1.45756	-3.2368	.
7	QUAD	FOSFX	1.5277	-2.6506	0.55160	-2.37666	-1.80041	-0.57389	7.75343	-1.51639	-2.5271	.
8	QUAD	FUSEX	-1.3724	-2.3057	-1.27919	1.13074	-1.17474	1.45756	-1.51639	3.28290	-2.1132	.
9	QUAD	_GIC_	3.4492	18.7299	2.39098	4.06890	6.14198	-3.23681	-2.52711	-2.11324	19.6353	.
10	eq	WGT	1.0000	1.0000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.0000	1.00000
11	LINEAR	_LINEAR	-0.4570	-0.0444	-0.18198	-0.30335	-0.99984	-0.31612	-0.94203	-0.45832	-0.5000	.
12	PARMS	WGT	.	.	.	.	.	.	.	.	.	.
13	ge	BONDS	0.0000	0.8603	0.00320	0.00030	0.00000	0.00000	0.01460	0.00000	0.8000	0.01647
14	le	BONDS	0.0000	0.8603	0.00320	0.00030	0.00000	0.00000	0.01460	0.00000	0.8000	0.21647
15	ge	CASH	0.0000	0.1397	0.05810	0.10570	0.19150	0.05610	0.08141	0.00830	0.2000	-0.07350
16	le	CASH	0.0000	0.1397	0.05810	0.10570	0.19150	0.05610	0.08141	0.00830	0.2000	0.12650
17	ge	FSTOCKS	0.0000	0.0000	0.18000	0.01570	0.13040	0.02770	0.78888	0.01930	0.0000	-0.05657
18	le	FSTOCKS	0.0000	0.0000	0.18000	0.01570	0.13040	0.02770	0.78888	0.01930	0.0000	0.10343
19	ge	HYBONDS	0.0000	0.0000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.0000	-0.10000
20	le	HYBONDS	0.0000	0.0000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.0000	0.10000
21	ge	OTHER	0.0000	0.0000	0.00540	0.03390	0.00480	0.00000	0.02940	0.00890	0.0000	-0.10000
22	le	OTHER	0.0000	0.0000	0.00540	0.03390	0.00480	0.00000	0.02940	0.00890	0.0000	0.10000
23	ge	STOCKS	1.0000	0.0000	0.93330	0.86010	0.80370	0.94390	0.87459	0.98280	0.0000	0.84703
24	le	STOCKS	1.0000	0.0000	0.93330	0.86010	0.80370	0.94390	0.87459	0.98280	0.0000	0.86703
25	lowerbd	LOWER	0.1000	0.0000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.0000	.
26	upperbd	UPPER	0.1000	1.0000	0.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.0000	.

Exhibit B

Obs	PRIORITY	VAR	TARGET	RANGE	MINWGT	MAXWGT
1	3	BONDS	0.15	0.10	0.05	0.25
2	3	CASH	0.00	0.10	0.00	0.10
3	3	FSTOCKS	0.15	0.08	0.07	0.23
4	3	HYBONDS	0.00	0.10	0.00	0.10
5	3	OTHER	0.00	0.10	0.00	0.10
6	3	STOCKS	0.85	0.01	0.84	0.86

Obs	CTICKER	TICKER	FNAME	RETURN
1	PF000021	CPO	CORN PRODUCTS INT'L	0.40227
2	PF000021	FBNDX	FIDELITY INVST GR BD	0.19281
3	PF000021	FCNTX	FIDELITY CONTRAFUND	-0.00948
4	PF000021	FGRIX	FIDELITY GROW & INC	-0.02961
5	PF000021	FLPSX	FIDELITY LOW PR STK	0.82059
6	PF000021	FMAGX	FIDELITY MAGELLAN	-0.08918
7	PF000021	FOSFX	FIDELITY OVERSEAS	-0.07460
8	PF000021	FUSEX	SPARTAN US EQ INDEX	-0.07849
9	PF000021	_GIC_	FID MGD INC PORT II	0.20503

Obs	TICKER	FNAME	STOCKS	DSTOCKS	FSTOCKS	BONDS	CASH	OTHER	SRC	ATYPE	DURABL
1	CPO	CORN PRODS INTL INC	1.0000	1.0000	0.0000	0.0000	0.0000	0.0000	FAME	EQUITY	0.0000
2	FBNDX	Fidelity Investment Grade Bond	0.0000	0.0000	0.0000	0.8603	0.1397	0.0000	MSTAR	BOND	.
3	FCNTX	Fidelity Contrafund	0.9333	0.7533	0.1800	0.0032	0.0581	0.0054	MSTAR	EQUITY	0.0293
4	FGRIX	Fidelity Growth & Income	0.8601	0.8444	0.0157	0.0003	0.1057	0.0339	MSTAR	EQUITY	0.0126
5	FLPSX	Fidelity Low-Priced Stock	0.8037	0.6733	0.1304	0.0000	0.1915	0.0048	MSTAR	EQUITY	0.0549
6	FMAGX	Fidelity Magellan	0.9439	0.9162	0.0277	0.0000	0.0561	0.0000	MSTAR	EQUITY	0.0134
7	FOSFX	Fidelity Overseas	0.8746	0.0857	0.7889	0.0146	0.0814	0.0294	MSTAR	FOREIGN	0.0298
8	FUSEX	Fidelity Spartan U.S. Equity Index	0.9828	0.9635	0.0193	0.0000	0.0083	0.0089	MSTAR	EQUITY	0.0185
9	_GIC_	Stable Value Fund	0.0000	0.0000	0.0000	0.8000	0.2000	0.0000		GIC	.

Obs	ENERGY	FINANC	HEALTH	INDUST	RETAIL	SERVIC	STAPLE	TECHNO	UTILIT	FOBJN	MCATN	HYBONDS	IGBONDS	ALPHA	PVALUEM
1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000			0	0.00000	-.001645568	-0.45699
2	.	.	.	.	.	.	.	.	.	Corp Bond--General	Intermediate-Term Bond	0	0.86030	-.000358043	-0.04444
3	0.0897	0.1785	0.1679	0.1228	0.0509	0.1347	0.1420	0.0791	0.0051	Growth	Large Blend	0	0.00320	-.002993691	-0.18198
4	0.0657	0.2187	0.1885	0.1202	0.0561	0.1119	0.0830	0.1392	0.0040	Growth and Income	Large Blend	0	0.00030	-.000884889	-0.30335
5	0.0489	0.1730	0.0941	0.1935	0.1154	0.1694	0.0485	0.0997	0.0028	Small Company	Small Value	0	0.00000	0.009368754	-0.99984
6	0.0763	0.2165	0.1547	0.1287	0.0745	0.1370	0.0590	0.1374	0.0025	Growth	Large Blend	0	0.00000	-.000592560	-0.31612
7	0.0751	0.2312	0.1326	0.0898	0.0332	0.1262	0.0413	0.2408	0.0000	Foreign Stock	Foreign Stock	0	0.01460	0.002990935	-0.94203
8	0.0718	0.1809	0.1377	0.1186	0.0669	0.1182	0.0727	0.1827	0.0320	Growth and Income	Large Blend	0	0.00000	-.000074559	-0.45832
9	.	.	.	.	.	.	.	.	.	Stable Value Fund	Stable Value Fund	0	0.80000	-.000985008	-0.50000

Obs	DATE	CPO	FBNDX	FCNTX	FGRIX	FLPSX	FMAGX	FOSFX	FUSEX	_GIC_	_BENCH	_EQBENCH
1	199904	0.20627	0.00333	0.02099	0.02487	0.07392	0.02389	0.04407	0.03840	0.003428	0.040139	0.046663
2	199905	0.02165	-0.01167	-0.02072	-0.02552	0.02718	-0.02875	-0.04531	-0.02396	-0.000800	-0.024338	-0.027080
3	199906	0.03461	-0.00349	0.05525	0.04863	0.04738	0.06517	0.05452	0.05531	0.002966	0.041641	0.049554
4	199907	0.03491	-0.00340	-0.01560	-0.03016	-0.01100	-0.03368	0.02906	-0.03111	0.002900	-0.018580	-0.021117
5	199908	0.03373	-0.00197	-0.02910	-0.01375	-0.03956	-0.01236	0.01400	-0.00528	0.002633	-0.006005	-0.006975
6	199909	-0.06224	0.01227	-0.00250	-0.02352	-0.02624	-0.01478	0.01331	-0.02755	0.006729	-0.014986	-0.019678
7	199910	0.06982	0.00098	0.03672	0.05080	-0.00970	0.06007	0.04208	0.06286	0.002959	0.050725	0.059025
8	199911	-0.03647	0.00082	0.04057	0.00949	0.02519	0.02049	0.07353	0.02019	0.002215	0.028682	0.033756
9	199912	0.04707	-0.00330	0.08919	0.04388	0.04194	0.07204	0.11787	0.05903	0.001031	0.065868	0.078343
10	200001	-0.29771	-0.00190	-0.03482	-0.04453	-0.02827	-0.04809	-0.05686	-0.05049	-0.000007	-0.039050	-0.045364
11	200002	0.02446	0.01102	0.04627	-0.02663	0.04273	0.01000	0.05543	-0.01901	0.006940	0.021558	0.023228
12	200003	0.02532	0.01275	0.04532	0.08085	0.03095	0.09059	0.01004	0.09769	0.005654	0.049395	0.055789
13	200004	-0.00260	-0.00476	-0.06471	-0.01966	-0.01226	-0.04314	-0.06049	-0.03012	0.001925	-0.044765	-0.052161
14	200005	0.02344	-0.00181	-0.03840	-0.01380	-0.00856	-0.03357	-0.03440	-0.02096	0.003612	-0.028150	-0.033037
15	200006	0.08300	0.02154	0.03868	0.03654	0.03023	0.03857	0.04430	0.02518	0.011032	0.039895	0.043265
16	200007	-0.05660	0.00841	-0.02148	-0.01015	0.00293	-0.01240	-0.04789	-0.01590	0.006847	-0.019182	-0.024169
17	200008	0.00500	0.01416	0.05310	0.04933	0.05892	0.06393	0.03950	0.06207	0.007924	0.054331	0.061361
18	200009	-0.09041	0.00679	-0.01866	-0.02460	0.00410	-0.05380	-0.06275	-0.05274	0.008238	-0.038995	-0.046986
19	200010	0.10440	0.00260	-0.02500	0.00558	0.00486	-0.01793	-0.04008	-0.00432	0.004475	-0.017396	-0.021632
20	200011	-0.03483	0.01682	-0.07290	-0.05304	-0.01584	-0.08513	-0.04249	-0.07868	0.009302	-0.072777	-0.088506
21	200012	0.20278	0.01815	0.03391	0.00856	0.06915	0.00838	0.00566	0.00473	0.012066	0.020631	0.020997
22	200101	-0.12258	0.01644	-0.03965	-0.01235	0.06661	0.03252	0.01688	0.03525	0.014167	0.029194	0.031462
23	200102	-0.00392	0.00910	-0.04658	-0.04930	-0.00649	-0.09368	-0.08097	-0.09121	0.007130	-0.076321	-0.091327
24	200103	0.01408	0.00380	-0.05174	-0.05122	-0.01918	-0.06396	-0.05044	-0.06350	0.008116	-0.056302	-0.067123
25	200104	-0.04521	-0.00616	0.04824	0.05827	0.06575	0.08756	0.08000	0.07738	0.003164	0.067505	0.080150
26	200105	0.16939	0.00759	0.01296	0.01339	0.03397	0.01045	-0.03066	0.00655	0.006162	0.002701	0.002112
27	200106	0.12069	0.00335	-0.02558	-0.01351	0.02455	-0.02467	-0.04479	-0.02206	0.003824	-0.017243	-0.020952
28	200107	-0.07187	0.02289	-0.00543	-0.00963	0.00332	-0.01530	-0.04000	-0.01242	0.012580	-0.010920	-0.016792
29	200108	0.11111	0.01139	-0.03004	-0.04578	-0.00404	-0.06481	-0.04098	-0.06265	0.006785	-0.044414	-0.054274
30	200109	-0.12633	0.01105	-0.05467	-0.04388	-0.07227	-0.08084	-0.11360	-0.08126	0.014888	-0.076294	-0.091813
31	200110	0.04769	0.02046	0.00794	0.00454	0.04091	0.02372	0.04379	0.01927	0.009985	0.024781	0.025461
32	200111	0.08140	-0.01183	0.03423	0.05202	0.06280	0.07480	0.05466	0.07641	-0.002699	0.057004	0.069497
33	200112	0.08597	-0.00519	0.02355	0.00726	0.05285	0.01043	0.00073	0.00874	0.000496	0.012519	0.015851
34	200201	-0.18667	0.00684	-0.00561	-0.01017	0.01605	-0.03147	-0.04923	-0.01501	0.002859	-0.015436	-0.019589
35	200202	0.07255	0.00900	0.00118	-0.00487	-0.00610	-0.01724	0.02186	-0.01924	0.004345	-0.011874	-0.015679
36	200203	0.05040	-0.01762	0.03335	0.02362	0.05923	0.03377	0.05180	0.03740	-0.006509	0.036330	0.045676

```
NOTE: Copyright (c) 1999-2001 by SAS Institute Inc., Cary, NC, USA.
NOTE: SAS (r) Proprietary Software Release 8.2 (TS2M0)
      Licensed to FIDELITY MANAGEMENT AND RESEARCH CO., Site 0003091043.
NOTE: This session is executing on the SunOS 5.6 platform.



This message is contained in the SAS news file, and is presented upon
initialization.  Edit the files "news" in the "misc/base" directory to
display site-specific news and information in the program log.
The command line option "-nonews" will prevent this display.



NOTE: SAS initialization used:
      real time           0.28 seconds
      cpu time            0.06 seconds

1          options nonotes nomprint validvarname=upcase noovp;
2
3          %let FPPCODE = %sysget(FPPCODE);
4          %let DSBASE=%sysget(DSBASE);
5          %include "&FPPCODE/utilities/macros.macro";

*** ANNOTATE macros are now available ***

 For further information on ANNOTATE macros, enter,
    %HELPANO(macroname), (for specific macros)
    %HELPANO(ALL), (for information on all macros)
 or %HELPANO (for a list of macro names)

9654       %include "&FPPCODE/utilities/dbutil.macro";
9818
9819       %include "&FPPCODE/pop/optimizer/getenv.macro";
9864       %include "&FPPCODE/pop/optimizer/target_adjuster.macro";
10191      %include "&FPPCODE/pop/optimizer/discopto.macro";
10836      %include "&FPPCODE/pop/optimizer/pop2_opto.macro";
11237      %include "&FPPCODE/pop/optimizer/mk_pfs.macro";
11542      %include "&FPPCODE/pop/optimizer/ctw2whole_pct.macro";
11636
11637      %include "&FPPCODE/fpp/special_cases.macro";
11785
11786      %getenv(makenew=1, inpf=1);
MPRINT(GETENV):  ;
MPRINT(GETENV):   libname product "/appdata/watermdl/fpp/PRODUCTION/200203";
NOTE: Libref PRODUCT was successfully assigned as follows:
      Engine:        V8
      Physical Name: /appdata/watermdl/fpp/PRODUCTION/200203
MPRINT(GETENV):   libname baseds "/appdata/watermdl/feed/ds/200203";
NOTE: Libref BASEDS was successfully assigned as follows:
      Engine:        V8
      Physical Name: /appdata/watermdl/feed/ds/200203
MPRINT(PFNAM2MAC):   data _null_;
MPRINT(PFNAM2MAC):   pfname = translate("FPP05003/2/10",' ','/');
MPRINT(PFNAM2MAC):   call symput("P", pfname);
MPRINT(PFNAM2MAC):   run;
```

Exhibit C

```
NOTE: DATA statement used:
      real time           0.00 seconds
      cpu time            0.01 seconds


pfname=FPP05003/2/10, p=FPP05003 2 10
MPRINT(PFNAM2MAC):  ;
MPRINT(PFNAM2MAC):   proc sql noprint;
MPRINT(PFNAM2MAC):   select pfid,pflname ,t1, put(100*w1,z2.) into :pfid, :pflname , :stktck1, :stkwgt1 from product.pfmaster where
pfname = "FPP05003/2/10";
MPRINT(GETENV):  ;
MPRINT(GETENV):   libname pf "/appdata/watermdl/fpp/PRODUCTION/200203/pf/FPP05003/2/10";
NOTE: Libref PF was successfully assigned as follows:
      Engine:        V8
      Physical Name: /appdata/watermdl/fpp/PRODUCTION/200203/pf/FPP05003/2/10
NOTE: PROCEDURE SQL used:
      real time           0.06 seconds
      cpu time            0.02 seconds


MPRINT(LIB2MA):   proc contents data=pf._all_ memtype=data noprint out=_l2ma;
MPRINT(LIB2MA):   run;

NOTE: The data set WORK._L2MA has 93 observations and 40 variables.
NOTE: PROCEDURE CONTENTS used:
      real time           0.21 seconds
      cpu time            0.04 seconds


MPRINT(LIB2MA):   data _l2ma;
MPRINT(LIB2MA):   set _l2ma(keep=memname);
MPRINT(LIB2MA):   by memname;
MPRINT(LIB2MA):   if first.memname;
MPRINT(LIB2MA):   run;

NOTE: There were 93 observations read from the data set WORK._L2MA.
NOTE: The data set WORK._L2MA has 14 observations and 1 variables.
NOTE: DATA statement used:
      real time           0.01 seconds
      cpu time            0.02 seconds


MPRINT(DS2MA):   proc sql noprint;
MPRINT(DS2MA):   select count(distinct memname) into :gds0 from _l2ma where memname ne '' ;
NOTE: PROCEDURE SQL used:
      real time           0.01 seconds
      cpu time            0.01 seconds


MPRINT(DS2MA):   proc sql noprint;
MPRINT(DS2MA):   select distinct memname into :gds1 - :gds14 from _l2ma where memname ne '' order by memname;
MPRINT(LIB2MA):  ;
NOTE: PROCEDURE SQL used:
      real time           0.00 seconds
      cpu time            0.01 seconds


MPRINT(TRASH):   proc datasets nolist lib=work;
```

```
MPRINT(TRASH):   delete _l2ma;
MPRINT(TRASH):   run;

NOTE: Deleting WORK._L2MA (memtype=DATA).
MPRINT(LIB2MA):  ;
MPRINT(TRASHLIB):  ;

NOTE: PROCEDURE DATASETS used:
      real time          0.03 seconds
      cpu time           0.03 seconds


MPRINT(TRASH):   proc datasets nolist lib=pf;
MPRINT(TRASH):   delete AACONST ADJCONST CTW INITDS INQUAD MDL_CTW MDL_DTT MDL_TERR OUTQUAD PLAST SCONST TAROUTQUAD TARQUAD TYCONST;
MPRINT(TRASH):   run;

NOTE: Deleting PF.AACONST (memtype=DATA).
NOTE: Deleting PF.ADJCONST (memtype=DATA).
NOTE: Deleting PF.CTW (memtype=DATA).
NOTE: Deleting PF.INITDS (memtype=DATA).
NOTE: Deleting PF.INQUAD (memtype=DATA).
NOTE: Deleting PF.MDL_CTW (memtype=DATA).
NOTE: Deleting PF.MDL_DTT (memtype=DATA).
NOTE: Deleting PF.MDL_TERR (memtype=DATA).
NOTE: Deleting PF.OUTQUAD (memtype=DATA).
NOTE: Deleting PF.PLAST (memtype=DATA).
NOTE: Deleting PF.SCONST (memtype=DATA).
NOTE: Deleting PF.TAROUTQUAD (memtype=DATA).
NOTE: Deleting PF.TARQUAD (memtype=DATA).
NOTE: Deleting PF.TYCONST (memtype=DATA).
11787
11788      %opto_pf;
Optimizing Portfolio PF000021 FPP05003/2/CPO/10

NOTE: PROCEDURE DATASETS used:
      real time          0.11 seconds
      cpu time           0.05 seconds


MPRINT(DSV2MA):   proc contents data=product.opop(drop = ) out=_m1(keep=name varnum ) short noprint;
MPRINT(DSV2MA):   run;

NOTE: The data set WORK._M1 has 5 observations and 2 variables.
NOTE: PROCEDURE CONTENTS used:
      real time          0.05 seconds
      cpu time           0.03 seconds


MPRINT(DS2MA):   proc sql noprint;
MPRINT(DS2MA):   select count(distinct name) into :_dm0 from _m1 where name ne '' ;
NOTE: PROCEDURE SQL used:
      real time          0.01 seconds
      cpu time           0.02 seconds


MPRINT(DS2MA):   proc sql noprint;
MPRINT(DS2MA):   select distinct name into :_dm1 - :_dm5 from _m1 where name ne '' order by name;
MPRINT(DSV2MA):  ;
```

```
NOTE: PROCEDURE SQL used:
      real time           0.00 seconds
      cpu time            0.01 seconds


MPRINT(TRASH):   proc datasets nolist lib=work;
MPRINT(TRASH):   delete _m1;
MPRINT(TRASH):   run;

NOTE: Deleting WORK._M1 (memtype=DATA).
MPRINT(DSV2MA):  ;
MPRINT(DS2MACRO):  ;

NOTE: PROCEDURE DATASETS used:
      real time           0.02 seconds
      cpu time            0.02 seconds


MPRINT(DS2MACRO):   proc sql noprint;
MPRINT(DS2MACRO):   select KAPPA,LAMBDA,LINEAR,MINVAL,WHOLE into :KAPPA, :LAMBDA, :LINEAR, :MINVAL, :WHOLE from product.opop;
MPRINT(OPTO_PF):  ;
NOTE: PROCEDURE SQL used:
      real time           0.01 seconds
      cpu time            0.01 seconds


MPRINT(OPTO_PF):   proc sql;
MPRINT(OPTO_PF):   create table pf.plast as select "PF000021" as cticker, ticker, fname from product.plast where cticker =
"FPP05003" order by ticker;
NOTE: Table PF.PLAST created, with 9 rows and 3 columns.

MPRINT(DBGETPFX):  ;
NOTE: PROCEDURE SQL used:
      real time           0.14 seconds
      cpu time            0.10 seconds


MPRINT(DBGETPFX):   proc sql noerrorstop;
MPRINT(DBGETPFX):   connect to sybase as pod( server=SYBCAUCHY database=pod user=wateradm pass=wateradm );
MPRINT(DBGETPFX):   create table pf.aaconst as select * from connection to pod(
MPRINT(GETAACON):   select 1 as priority, var,target,range from pop2_aaconst where RUNID='fpp/PRODUCTION/200203' and
PFNAME='FPP05003/2/10' union select 2 as priority, var,target,range from pop2_aaconst where RUNID='fpp/PRODUCTION/200203' and
PFNAME='FPP05003/2' union select 3 as priority, var,target,range from pop2_aaconst where RUNID='fpp/PRODUCTION/DEFAULT' and
PFNAME='DEFAULT/2' order by var, priority );
NOTE: Table PF.AACONST created, with 6 rows and 4 columns.

NOTE: PROCEDURE SQL used:
      real time           4.78 seconds
      cpu time            0.17 seconds


MPRINT(GETAACON):   data pf.aaconst(drop=irange);
MPRINT(GETAACON):   set pf.aaconst(rename=(range=irange));
MPRINT(GETAACON):   retain range;
MPRINT(GETAACON):   by var priority;
MPRINT(GETAACON):   if first.var then range = irange;
MPRINT(GETAACON):   minwgt = target - range;
MPRINT(GETAACON):   if minwgt < 0 then minwgt = 0;
```

```
MPRINT(GETAACON):   maxwgt = target + range;
MPRINT(GETAACON):   if maxwgt > 1 then maxwgt = 1;
MPRINT(GETAACON):   if last.var then output;
MPRINT(GETAACON):   run;

NOTE: There were 6 observations read from the data set PF.AACONST.
NOTE: The data set PF.AACONST has 6 observations and 6 variables.
NOTE: DATA statement used:
      real time           0.03 seconds
      cpu time            0.01 seconds


MPRINT(OPTO_PF):  ;
MPRINT(DS2MA):   proc sql noprint;
MPRINT(DS2MA):   select count(distinct ticker) into :_mz0 from pf.plast where ticker ne '' ;
NOTE: PROCEDURE SQL used:
      real time           0.02 seconds
      cpu time            0.00 seconds


MPRINT(DS2MA):   proc sql noprint;
MPRINT(DS2MA):   select distinct ticker into :_mz1 - :_mz9 from pf.plast where ticker ne '' order by ticker;
MPRINT(DS2L):  ;
MPRINT(MK_BENDTT):  ;
NOTE: PROCEDURE SQL used:
      real time           0.01 seconds
      cpu time            0.01 seconds


MPRINT(MK_BENDTT):   proc transpose data=pf.aaconst(keep=var target) out=fpp_targ(drop=_name_);
MPRINT(MK_BENDTT):   id var;
MPRINT(MK_BENDTT):   var target;
MPRINT(MK_BENDTT):   run;

NOTE: There were 6 observations read from the data set PF.AACONST.
NOTE: The data set WORK.FPP_TARG has 1 observations and 7 variables.
NOTE: PROCEDURE TRANSPOSE used:
      real time           0.02 seconds
      cpu time            0.02 seconds


MPRINT(MK_BENDTT):   data fpp_targ(rename=(stocks=stocks_ fstocks=fstocks_ bonds=bonds_ cash=cash_ hybonds=hybonds_));
MPRINT(MK_BENDTT):   format stocks fstocks bonds hybonds cash 8.4;
MPRINT(MK_BENDTT):   set fpp_targ end=eof;
MPRINT(MK_BENDTT):   if eof then do;
MPRINT(MK_BENDTT):   call symput("TAR_FSTOCKS",compress(put(fstocks,best.)));
MPRINT(MK_BENDTT):   call symput("TAR_DSTOCKS",compress(put( stocks-fstocks,best.)));
MPRINT(MK_BENDTT):   call symput("TAR_STOCKS",compress(put( stocks,best.)));
MPRINT(MK_BENDTT):   end;
MPRINT(MK_BENDTT):   run;

NOTE: There were 1 observations read from the data set WORK.FPP_TARG.
NOTE: The data set WORK.FPP_TARG has 1 observations and 7 variables.
NOTE: DATA statement used:
      real time           0.02 seconds
      cpu time            0.01 seconds
```

```
MPRINT(MK_BENDTT):   data pf.mdl_dtt(keep=date _bench _EQBENCH CPO FBNDX FCNTX FGRIX FLPSX FMAGX FOSFX FUSEX _GIC_);
MPRINT(MK_BENDTT):   merge fpp_targ product.dtt;
MPRINT(MK_BENDTT):   retain Tstocks_ Tfstocks_ Tbonds_ Thybonds_ Tcash_;
MPRINT(MK_BENDTT):   array _ac(*) stocks_ fstocks_ bonds_ hybonds_ cash_;
MPRINT(MK_BENDTT):   array _tac(*) Tstocks_ Tfstocks_ Tbonds_ Thybonds_ Tcash_;
MPRINT(MK_BENDTT):   if _n_ eq 1 then do;
MPRINT(MK_BENDTT):   do i = 1 to 5;
MPRINT(MK_BENDTT):   if _ac[i] eq . then _ac[i] = 0;
MPRINT(MK_BENDTT):   _tac[i] = _ac[i];
MPRINT(MK_BENDTT):   end;
MPRINT(MK_BENDTT):   end;
MPRINT(MK_BENDTT):   _bench = (tstocks_-tfstocks_)*_w5000 + tfstocks_*_eafe + (tbonds_-thybonds_)*_bond +thybonds_*_ml_hym +
tcash_*_cash_ ;
MPRINT(MK_BENDTT):   if tstocks_ > 0 then _EQBENCH=((tstocks_-tfstocks_)*_w5000 + tfstocks_*_eafe)/tstocks_;
MPRINT(MK_BENDTT):   else _eqbench = .;
MPRINT(MK_BENDTT):   run;

NOTE: There were 1 observations read from the data set WORK.FPP_TARG.
NOTE: There were 36 observations read from the data set PRODUCT.DTT.
NOTE: The data set PF.MDL_DTT has 36 observations and 12 variables.
NOTE: DATA statement used:
      real time           0.37 seconds
      cpu time            0.32 seconds


MPRINT(OPTO_PF):   ;
MPRINT(OPTO_PF):   ;
MPRINT(DBGETPFX):   ;
MPRINT(DBGETPFX):   proc sql noerrorstop;
MPRINT(DBGETPFX):   connect to sybase as pod( server=SYBCAUCHY database=pod user=wateradm pass=wateradm );
MPRINT(DBGETPFX):   create table initds as select * from connection to pod(
MPRINT(GETOPINI):   select ticker, wgt from pop2_ini where RUNID='fpp/PRODUCTION/200203' and PFNAME = 'FPP05003/2/10' order by
ticker );
NOTE: Table WORK.INITDS created, with 0 rows and 2 columns.

NOTE: PROCEDURE SQL used:
      real time           0.57 seconds
      cpu time            0.10 seconds


MPRINT(GETOPINI):   data pf.initds;
MPRINT(GETOPINI):   merge pf.plast initds;
MPRINT(GETOPINI):   by ticker;
MPRINT(GETOPINI):   if wgt eq . then wgt = 0;
MPRINT(GETOPINI):   run;

NOTE: There were 9 observations read from the data set PF.PLAST.
NOTE: There were 0 observations read from the data set WORK.INITDS.
NOTE: The data set PF.INITDS has 9 observations and 4 variables.
NOTE: DATA statement used:
      real time           0.02 seconds
      cpu time            0.01 seconds


MPRINT(OPTO_PF):   ;
MPRINT(DBGETPFX):   ;
MPRINT(DBGETPFX):   proc sql noerrorstop;
MPRINT(DBGETPFX):   connect to sybase as pod( server=SYBCAUCHY database=pod user=wateradm pass=wateradm );
```

```
MPRINT(DBGETPFX):   create table _const as select * from connection to pod(
MPRINT(GETTYCON):   select 1 as priority, typevar,typeval,minwgt,maxwgt from pop2_tyconst where RUNID='fpp/PRODUCTION/200203' and
PFNAME='FPP05003/2/10' union select 2 as priority, typevar,typeval,minwgt,maxwgt from pop2_tyconst where
RUNID='fpp/PRODUCTION/200203' and PFNAME='FPP05003/2' union select 3 as priority, typevar,typeval,minwgt,maxwgt from pop2_tyconst
where RUNID='fpp/PRODUCTION/200203' and PFNAME='' order by typevar,typeval,priority );
NOTE: Table WORK._CONST created, with 0 rows and 5 columns.

NOTE: PROCEDURE SQL used:
      real time           0.12 seconds
      cpu time            0.09 seconds


MPRINT(GETTYCON):   data _const;
MPRINT(GETTYCON):   set _const;
MPRINT(GETTYCON):   by typevar typeval priority;
MPRINT(GETTYCON):   if first.priority;
MPRINT(GETTYCON):   run;

NOTE: There were 0 observations read from the data set WORK._CONST.
NOTE: The data set WORK._CONST has 0 observations and 5 variables.
NOTE: DATA statement used:
      real time           0.01 seconds
      cpu time            0.01 seconds


MPRINT(DS2MA):   proc sql noprint;
MPRINT(DS2MA):   select count(*) into :_tvr0 from _const ;
MPRINT(GETTYCON):  ;
NOTE: PROCEDURE SQL used:
      real time           0.00 seconds
      cpu time            0.01 seconds


MPRINT(DS2MA):   proc sql noprint;
MPRINT(DS2MA):   select count(*) into :_tvl0 from _const ;
MPRINT(GETTYCON):  ;
NOTE: PROCEDURE SQL used:
      real time           0.00 seconds
      cpu time            0.00 seconds


MPRINT(DS2MA):   proc sql noprint;
MPRINT(DS2MA):   select count(*) into :_tmn0 from _const ;
MPRINT(GETTYCON):  ;
NOTE: PROCEDURE SQL used:
      real time           0.00 seconds
      cpu time            0.01 seconds


MPRINT(DS2MA):   proc sql noprint;
MPRINT(DS2MA):   select count(*) into :_tmx0 from _const ;
MPRINT(GETTYCON):  ;
NOTE: PROCEDURE SQL used:
      real time           0.00 seconds
      cpu time            0.01 seconds


MPRINT(GETTYCON):   proc sql noprint;
```

```
MPRINT(GETTYCON):   create table pf.tyconst(ticker char(8), minwgt num, maxwgt num);
NOTE: Table PF.TYCONST created, with 0 rows and 3 columns.
NOTE: PROCEDURE SQL used:
      real time           0.02 seconds
      cpu time            0.00 seconds


MPRINT(GETTYCON):   proc sort data=pf.tyconst;
MPRINT(GETTYCON):   by ticker;
MPRINT(GETTYCON):   run;

NOTE: Input data set is empty.
NOTE: The data set PF.TYCONST has 0 observations and 3 variables.
NOTE: PROCEDURE SORT used:
      real time           0.02 seconds
      cpu time            0.02 seconds


MPRINT(TY_SPECIAL_CASES):   data gicon;
MPRINT(TY_SPECIAL_CASES):   length ticker $8.;
MPRINT(TY_SPECIAL_CASES):   ticker = '_GIC_';
MPRINT(TY_SPECIAL_CASES):   minwgt = 0;
MPRINT(TY_SPECIAL_CASES):   maxwgt = 1;
MPRINT(TY_SPECIAL_CASES):   output;
MPRINT(TY_SPECIAL_CASES):   ticker = '_0632';
MPRINT(TY_SPECIAL_CASES):   output;
MPRINT(TY_SPECIAL_CASES):   ticker = '_0633';
MPRINT(TY_SPECIAL_CASES):   output;
MPRINT(TY_SPECIAL_CASES):   run;

NOTE: The data set WORK.GICON has 3 observations and 3 variables.
NOTE: DATA statement used:
      real time           0.04 seconds
      cpu time            0.01 seconds


MPRINT(TY_SPECIAL_CASES):   proc append base=pf.tyconst data=gicon;
NOTE: Appending WORK.GICON to PF.TYCONST.
NOTE: There were 3 observations read from the data set WORK.GICON.
NOTE: 3 observations added.
NOTE: The data set PF.TYCONST has 3 observations and 3 variables.
NOTE: PROCEDURE APPEND used:
      real time           0.02 seconds
      cpu time            0.01 seconds


MPRINT(TY_SPECIAL_CASES):   proc sort data=pf.tyconst;
MPRINT(TY_SPECIAL_CASES):   by ticker;
MPRINT(TY_SPECIAL_CASES):   run;

NOTE: There were 3 observations read from the data set PF.TYCONST.
NOTE: The data set PF.TYCONST has 3 observations and 3 variables.
NOTE: PROCEDURE SORT used:
      real time           0.01 seconds
      cpu time            0.01 seconds


MPRINT(GETTYCON):   ;
```

```
MPRINT(OPTO_PF):  ;
MPRINT(GETFACON):   data faconst;
MPRINT(GETFACON):   format ticker $8. wgt best.;
MPRINT(GETFACON):   if 1 > 0;
MPRINT(GETFACON):   ticker = "CPO     ";
MPRINT(GETFACON):   wgt = 10/100;
MPRINT(GETFACON):   minwgt=wgt;
MPRINT(GETFACON):   maxwgt=wgt;
MPRINT(GETFACON):   if ticker ne '' then output;
MPRINT(GETFACON):   run;

NOTE: The data set WORK.FACONST has 1 observations and 4 variables.
NOTE: DATA statement used:
      real time          0.01 seconds
      cpu time           0.00 seconds


MPRINT(OPTO_PF):  ;
MPRINT(OPTO_PF):   data pf.sconst;
MPRINT(OPTO_PF):   merge pf.tyconst faconst;
MPRINT(OPTO_PF):   by ticker;
MPRINT(OPTO_PF):   if wgt ne . then do;
MPRINT(OPTO_PF):   maxwgt=wgt;
MPRINT(OPTO_PF):   minwgt=wgt;
MPRINT(OPTO_PF):   end;
MPRINT(OPTO_PF):   run;

NOTE: There were 3 observations read from the data set PF.TYCONST.
NOTE: There were 1 observations read from the data set WORK.FACONST.
NOTE: The data set PF.SCONST has 4 observations and 4 variables.
NOTE: DATA statement used:
      real time          0.01 seconds
      cpu time           0.01 seconds


MPRINT(CLEANRET):   data _tr1;
MPRINT(CLEANRET):   set pf.mdl_dtt;
MPRINT(CLEANRET):   array __x _numeric_;
MPRINT(CLEANRET):   do over __x;
MPRINT(CLEANRET):   if __x = . then __x = 0;
MPRINT(CLEANRET):   end;
MPRINT(CLEANRET):   output;
MPRINT(CLEANRET):   run;

NOTE: There were 36 observations read from the data set PF.MDL_DTT.
NOTE: The data set WORK._TR1 has 36 observations and 12 variables.
NOTE: DATA statement used:
      real time          0.01 seconds
      cpu time           0.01 seconds


MPRINT(TRASH):   proc datasets nolist lib=work;
MPRINT(TRASH):   delete _rc3;
MPRINT(TRASH):   run;

NOTE: The file WORK._RC3 (memtype=DATA) was not found, but appears on a DELETE statement.
MPRINT(CLEANRET):  ;
MPRINT(TOTALRET):  ;
```

```
NOTE: PROCEDURE DATASETS used:
      real time           0.02 seconds
      cpu time            0.00 seconds

MPRINT(DSV2FL):   proc contents data=_tr1(drop = DATE) out=_m1(keep=name varnum) short noprint;
MPRINT(DSV2FL):   run;

NOTE: The data set WORK._M1 has 11 observations and 2 variables.
NOTE: PROCEDURE CONTENTS used:
      real time           0.02 seconds
      cpu time            0.00 seconds


MPRINT(DSV2FL):   proc sort data=_m1;
MPRINT(DSV2FL):   by varnum;
MPRINT(DSV2FL):   run;

NOTE: There were 11 observations read from the data set WORK._M1.
NOTE: The data set WORK._M1 has 11 observations and 2 variables.
NOTE: PROCEDURE SORT used:
      real time           0.01 seconds
      cpu time            0.00 seconds


MPRINT(DSV2FL):   data _null_;
MPRINT(DSV2FL):   set _m1 end=eof;
MPRINT(DSV2FL):   if _n_ = 1 then call symput("_f",name);
MPRINT(DSV2FL):   if eof then do;
MPRINT(DSV2FL):   call symput("_l",name);
MPRINT(DSV2FL):   call symput("_c",trim(left(put(_n_,8.))));
MPRINT(DSV2FL):   end;
MPRINT(DSV2FL):   run;

NOTE: There were 11 observations read from the data set WORK._M1.
NOTE: DATA statement used:
      real time           0.00 seconds
      cpu time            0.01 seconds


MPRINT(TRASH):   proc datasets nolist lib=work;
MPRINT(TRASH):   delete _m1;
MPRINT(TRASH):   run;

NOTE: Deleting WORK._M1 (memtype=DATA).
MPRINT(DSV2FL):  ;
MPRINT(TOTALRET):  ;
NOTE: PROCEDURE DATASETS used:
      real time           0.02 seconds
      cpu time            0.00 seconds


MPRINT(TOTALRET):   data _ret(drop = __x1 - __x11 _i );
MPRINT(TOTALRET):   set _tr1 end=eof;
MPRINT(TOTALRET):   array __t[11] CPO -- _EQBENCH ;
MPRINT(TOTALRET):   array __x[11];
MPRINT(TOTALRET):   retain __x1 - __x11 1;
```

```
MPRINT(TOTALRET):   do _i = 1 to 11;
MPRINT(TOTALRET):   __x[_i] = __x[_i] * (1+ __t[_i]);
MPRINT(TOTALRET):   end;
MPRINT(TOTALRET):   if eof then do;
MPRINT(TOTALRET):   do _i = 1 to 11;
MPRINT(TOTALRET):   __t[_i] = __x[_i] - 1;
MPRINT(TOTALRET):   end;
MPRINT(TOTALRET):   output;
MPRINT(TOTALRET):   end;
MPRINT(TOTALRET):   run;

NOTE: There were 36 observations read from the data set WORK._TR1.
NOTE: The data set WORK._RET has 1 observations and 12 variables.
NOTE: DATA statement used:
      real time          0.02 seconds
      cpu time           0.02 seconds


MPRINT(TRASH):   proc datasets nolist lib=work;
MPRINT(TRASH):   delete _tr1 _tr2;
MPRINT(TRASH):   run;

NOTE: The file WORK._TR2 (memtype=DATA) was not found, but appears on a DELETE statement.
NOTE: Deleting WORK._TR1 (memtype=DATA).
MPRINT(TOTALRET):  ;
MPRINT(RET2TV):  ;

NOTE: PROCEDURE DATASETS used:
      real time          0.02 seconds
      cpu time           0.00 seconds


MPRINT(RET2TV):   proc transpose data= _ret (drop=date) out=_a4(rename=(col1=return)) name=ticker;

NOTE: There were 1 observations read from the data set WORK._RET.
NOTE: The data set WORK._A4 has 11 observations and 2 variables.
NOTE: PROCEDURE TRANSPOSE used:
      real time          0.02 seconds
      cpu time           0.02 seconds


MPRINT(RET2TV):   proc sort data=_a4;
MPRINT(RET2TV):   by ticker;

NOTE: There were 11 observations read from the data set WORK._A4.
NOTE: The data set WORK._A4 has 11 observations and 2 variables.
NOTE: PROCEDURE SORT used:
      real time          0.01 seconds
      cpu time           0.02 seconds


MPRINT(RET2TV):   data pf.plast;
MPRINT(RET2TV):   merge pf.plast(in=int) _a4(keep=ticker return);
MPRINT(RET2TV):   by ticker;
MPRINT(RET2TV):   if int;
MPRINT(RET2TV):   run;

NOTE: There were 9 observations read from the data set PF.PLAST.
```

```
NOTE: There were 11 observations read from the data set WORK._A4.
NOTE: The data set PF.PLAST has 9 observations and 4 variables.
NOTE: DATA statement used:
      real time           0.02 seconds
      cpu time            0.02 seconds


MPRINT(TRASH):   proc datasets nolist lib=work;
MPRINT(TRASH):   delete _a4;
MPRINT(TRASH):   run;

NOTE: Deleting WORK._A4 (memtype=DATA).
MPRINT(RET2TV):  ;
MPRINT(OPTO_PF):  ;

NOTE: PROCEDURE DATASETS used:
      real time           0.02 seconds
      cpu time            0.01 seconds


MPRINT(TARADJ):   data pf.adjconst;
MPRINT(TARADJ):   set pf.aaconst;
MPRINT(TARADJ):   minwgt=target-range;
MPRINT(TARADJ):   maxwgt=target+range;
MPRINT(TARADJ):   run;

NOTE: There were 6 observations read from the data set PF.AACONST.
NOTE: The data set PF.ADJCONST has 6 observations and 6 variables.
NOTE: DATA statement used:
      real time           0.01 seconds
      cpu time            0.01 seconds


MPRINT(TARADJ):   proc sql noprint;
MPRINT(TARADJ):   select 1-sum(wgt) into :newmaxwt from faconst;
NOTE: PROCEDURE SQL used:
      real time           0.00 seconds
      cpu time            0.00 seconds


MPRINT(TARADJ):   proc transpose data=pf.aaconst(keep=var target) out=targets(drop=_label_);
MPRINT(TARADJ):   var target;
MPRINT(TARADJ):   id var;
MPRINT(TARADJ):   run;

NOTE: There were 6 observations read from the data set PF.AACONST.
NOTE: The data set WORK.TARGETS has 1 observations and 7 variables.
NOTE: PROCEDURE TRANSPOSE used:
      real time           0.01 seconds
      cpu time            0.00 seconds


MPRINT(TARADJ):   data targets;
MPRINT(TARADJ):   format other 8.4;
MPRINT(TARADJ):   set targets;
MPRINT(TARADJ):   DSTOCKS=STOCKS-FSTOCKS;
MPRINT(TARADJ):   IGBONDS=BONDS-HYBONDS;
MPRINT(TARADJ):   call symput("_EQTOT",compress(put(stocks*1.05,10.6)));
```

```
MPRINT(TARADJ):   run;

NOTE: There were 1 observations read from the data set WORK.TARGETS.
NOTE: The data set WORK.TARGETS has 1 observations and 9 variables.
NOTE: DATA statement used:
      real time           0.01 seconds
      cpu time            0.01 seconds


MPRINT(TARADJ):   proc transpose data=targets(drop=STOCKS BONDS OTHER) out=targets(rename=(_name_=var));
MPRINT(TARADJ):   run;

NOTE: There were 1 observations read from the data set WORK.TARGETS.
NOTE: The data set WORK.TARGETS has 5 observations and 2 variables.
NOTE: PROCEDURE TRANSPOSE used:
      real time           0.01 seconds
      cpu time            0.02 seconds


MPRINT(VAR2TICKER):   proc sort data=targets out=targets;
MPRINT(VAR2TICKER):   by var;
NOTE: There were 5 observations read from the data set WORK.TARGETS.
NOTE: The data set WORK.TARGETS has 5 observations and 2 variables.
NOTE: PROCEDURE SORT used:
      real time           0.01 seconds
      cpu time            0.00 seconds


MPRINT(VAR2TICKER):   data targets;
MPRINT(VAR2TICKER):   length ticker $8;
MPRINT(VAR2TICKER):   merge targets(in=ina) ast_idx(keep=var ticker in=inb);
MPRINT(VAR2TICKER):   by var;
MPRINT(VAR2TICKER):   if ina and inb;
MPRINT(VAR2TICKER):   run;

NOTE: There were 5 observations read from the data set WORK.TARGETS.
NOTE: There were 5 observations read from the data set WORK.AST_IDX.
NOTE: The data set WORK.TARGETS has 5 observations and 3 variables.
NOTE: DATA statement used:
      real time           0.02 seconds
      cpu time            0.01 seconds


MPRINT(TARADJ):  ;
MPRINT(TARADJ):   data aarange;
MPRINT(TARADJ):   set targets(keep=var ticker target) end=eof;
MPRINT(TARADJ):   if ticker eq '_BOND' then minwgt=target*.75;
MPRINT(TARADJ):   else if ticker eq '_CASH_' then minwgt = target/2;
MPRINT(TARADJ):   if ticker eq '_W5000' then maxwgt = target*1.2 + 10/100;
MPRINT(TARADJ):   else if ticker eq "_EAFE" then maxwgt = target*1.2;
MPRINT(TARADJ):   else if ticker eq "_ML_HYM" then maxwgt = .1;
MPRINT(TARADJ):   if minwgt eq . then minwgt = 0;
MPRINT(TARADJ):   if maxwgt eq . then maxwgt = 1;
MPRINT(TARADJ):   output;
MPRINT(TARADJ):   if eof then do;
MPRINT(TARADJ):   var='STOCKS';
MPRINT(TARADJ):   ticker = '';
MPRINT(TARADJ):   minwgt = 0;
```

```
MPRINT(TARADJ):   maxwgt = 0.892500;
MPRINT(TARADJ):   output;
MPRINT(TARADJ):   var='BONDS';
MPRINT(TARADJ):   ticker = '';
MPRINT(TARADJ):   minwgt = 0;
MPRINT(TARADJ):   maxwgt = 1;
MPRINT(TARADJ):   output;
MPRINT(TARADJ):   end;
MPRINT(TARADJ):   run;

NOTE: There were 5 observations read from the data set WORK.TARGETS.
NOTE: The data set WORK.AARANGE has 7 observations and 5 variables.
NOTE: DATA statement used:
      real time          0.02 seconds
      cpu time           0.01 seconds


MPRINT(TARADJ):   proc sort data=aarange;
MPRINT(TARADJ):   by ticker;
MPRINT(TARADJ):   run;

NOTE: There were 7 observations read from the data set WORK.AARANGE.
NOTE: The data set WORK.AARANGE has 7 observations and 5 variables.
NOTE: PROCEDURE SORT used:
      real time          0.01 seconds
      cpu time           0.00 seconds


MPRINT(TARADJ):   data quad_tv(keep=ticker stocks dstocks fstocks igbonds bonds cash hybonds);
MPRINT(TARADJ):   merge aarange(in=ina where=(ticker ne '')) faconst(in=inb) product.asset_tv(in=inc keep=ticker stocks dstocks
fstocks bonds igbonds cash hybonds) ast_idx(in=ind);
MPRINT(TARADJ):   by ticker;
MPRINT(TARADJ):   if inb or ind;
MPRINT(TARADJ):   run;

NOTE: There were 5 observations read from the data set WORK.AARANGE.
      WHERE ticker not = '';
NOTE: There were 1 observations read from the data set WORK.FACONST.
NOTE: There were 968 observations read from the data set PRODUCT.ASSET_TV.
NOTE: There were 5 observations read from the data set WORK.AST_IDX.
NOTE: The data set WORK.QUAD_TV has 6 observations and 8 variables.
NOTE: DATA statement used:
      real time          0.06 seconds
      cpu time           0.03 seconds


MPRINT(DS2MA):   proc sql noprint;
MPRINT(DS2MA):   select count(distinct ticker) into :_mz0 from quad_tv where ticker ne '' ;
NOTE: PROCEDURE SQL used:
      real time          0.02 seconds
      cpu time           0.00 seconds


MPRINT(DS2MA):   proc sql noprint;
MPRINT(DS2MA):   select distinct ticker into :_mz1 - :_mz6 from quad_tv where ticker ne '' order by ticker;
MPRINT(DS2L):  ;
MPRINT(TARADJ):  ;
TARADJ: optimizing CPO _BOND _CASH_ _EAFE _ML_HYM _W5000
```

```
NOTE: PROCEDURE SQL used:
      real time           0.01 seconds
      cpu time            0.01 seconds


MPRINT(TARADJ):   data _tadtt;
MPRINT(TARADJ):   merge product.dtt( keep = date CPO _BOND _CASH_ _EAFE _ML_HYM _W5000) pf.mdl_dtt(keep=date _bench _eqbench );
MPRINT(TARADJ):   by date;
MPRINT(TARADJ):   run;

NOTE: There were 36 observations read from the data set PRODUCT.DTT.
NOTE: There were 36 observations read from the data set PF.MDL_DTT.
NOTE: The data set WORK._TADTT has 36 observations and 9 variables.
NOTE: DATA statement used:
      real time           0.04 seconds
      cpu time            0.03 seconds


MPRINT(TARADJ):   data _cvin(drop=_bench);
MPRINT(TARADJ):   set _tadtt(keep=CPO _BOND _CASH_ _EAFE _ML_HYM _W5000 _bench);
MPRINT(TARADJ):   array _x CPO _BOND _CASH_ _EAFE _ML_HYM _W5000;
MPRINT(TARADJ):   do over _x;
MPRINT(TARADJ):   _x = (_x-_bench)*100;
MPRINT(TARADJ):   end;
MPRINT(TARADJ):   run;

NOTE: There were 36 observations read from the data set WORK._TADTT.
NOTE: The data set WORK._CVIN has 36 observations and 6 variables.
NOTE: DATA statement used:
      real time           0.02 seconds
      cpu time            0.01 seconds


MPRINT(TARADJ):   proc corr cov data=_cvin outp=_cov nosimple noprint;
MPRINT(TARADJ):   var CPO _BOND _CASH_ _EAFE _ML_HYM _W5000;
MPRINT(TARADJ):   run;

NOTE: The data set WORK._COV has 15 observations and 8 variables.
NOTE: PROCEDURE CORR used:
      real time           0.01 seconds
      cpu time            0.02 seconds


MPRINT(DS2MA):   proc sql noprint;
MPRINT(DS2MA):   select count(distinct ticker) into :al0 from quad_tv where ticker ne '' ;
NOTE: PROCEDURE SQL used:
      real time           0.02 seconds
      cpu time            0.01 seconds


MPRINT(DS2MA):   proc sql noprint;
MPRINT(DS2MA):   select distinct ticker into :al1 - :al6 from quad_tv where ticker ne '' order by ticker;
MPRINT(QUAD):  ;
NOTE: PROCEDURE SQL used:
      real time           0.00 seconds
      cpu time            0.01 seconds
```

```
MPRINT(QUAD):   data pf.tarquad(keep = _type_ _name_ CPO _BOND _CASH_ _EAFE _ML_HYM _W5000 _rhs_);
MPRINT(QUAD):   format _type_ $8. _name_ $8. _vname $8.;
MPRINT(QUAD):   set _cov(where=(_type_ = 'COV' and _name_ in ("CPO" ,"_BOND" ,"_CASH_" ,"_EAFE" ,"_ML_HYM" ,"_W5000")) ) end = eof;
MPRINT(QUAD):   retain _rhs_ . _vname '';
MPRINT(QUAD):   _type_ = 'QUAD';
MPRINT(QUAD):   CPO = CPO * 2 * 1;
MPRINT(QUAD):   _BOND = _BOND * 2 * 1;
MPRINT(QUAD):   _CASH_ = _CASH_ * 2 * 1;
MPRINT(QUAD):   _EAFE = _EAFE * 2 * 1;
MPRINT(QUAD):   _ML_HYM = _ML_HYM * 2 * 1;
MPRINT(QUAD):   _W5000 = _W5000 * 2 * 1;
MPRINT(QUAD):   output;
MPRINT(QUAD):   if eof then do;
MPRINT(QUAD):   CPO = 1;
MPRINT(QUAD):   _BOND = 1;
MPRINT(QUAD):   _CASH_ = 1;
MPRINT(QUAD):   _EAFE = 1;
MPRINT(QUAD):   _ML_HYM = 1;
MPRINT(QUAD):   _W5000 = 1;
MPRINT(QUAD):   _type_ = 'eq';
MPRINT(QUAD):   _name_ = 'WGT';
MPRINT(QUAD):   _rhs_ = 1;
MPRINT(QUAD):   output;
MPRINT(QUAD):   end;
MPRINT(QUAD):   run;

NOTE: There were 6 observations read from the data set WORK._COV.
      WHERE (_type_='COV') and _name_ in ('CPO', '_BOND', '_CASH_', '_EAFE', '_ML_HYM', '_W5000');
NOTE: The data set PF.TARQUAD has 7 observations and 9 variables.
NOTE: DATA statement used:
      real time          0.04 seconds
      cpu time           0.03 seconds


MPRINT(S2QCONST):   proc sort data=faconst(keep = ticker maxwgt) out=_q3;
MPRINT(S2QCONST):   by ticker maxwgt;

NOTE: There were 1 observations read from the data set WORK.FACONST.
NOTE: The data set WORK._Q3 has 1 observations and 2 variables.
NOTE: PROCEDURE SORT used:
      real time          0.03 seconds
      cpu time           0.01 seconds


MPRINT(S2QCONST):   data _q3;
MPRINT(S2QCONST):   set _q3;
MPRINT(S2QCONST):   by ticker maxwgt;
MPRINT(S2QCONST):   if first.ticker;

NOTE: There were 1 observations read from the data set WORK._Q3.
NOTE: The data set WORK._Q3 has 1 observations and 2 variables.
NOTE: DATA statement used:
      real time          0.02 seconds
      cpu time           0.02 seconds


MPRINT(S2QCONST):   proc transpose data=_q3 out=_q4;
MPRINT(S2QCONST):   var maxwgt;
```

```
MPRINT(S2QCONST):   id ticker;
NOTE: There were 1 observations read from the data set WORK._Q3.
NOTE: The data set WORK._Q4 has 1 observations and 2 variables.
NOTE: PROCEDURE TRANSPOSE used:
      real time             0.01 seconds
      cpu time              0.01 seconds


MPRINT(S2QCONST):   data _q4;
MPRINT(S2QCONST):   format _type_ $8. _name_ $8.;
MPRINT(S2QCONST):   set _q4;
MPRINT(S2QCONST):   _type_ = 'upperbd ';
MPRINT(S2QCONST):   _name_ = 'UPPER';
MPRINT(S2QCONST):   _RHS_ = .;
MPRINT(S2QCONST):   if CPO eq . then CPO = 1;
MPRINT(S2QCONST):   if _BOND eq . then _BOND = 1;
MPRINT(S2QCONST):   if _CASH_ eq . then _CASH_ = 1;
MPRINT(S2QCONST):   if _EAFE eq . then _EAFE = 1;
MPRINT(S2QCONST):   if _ML_HYM eq . then _ML_HYM = 1;
MPRINT(S2QCONST):   if _W5000 eq . then _W5000 = 1;
MPRINT(S2QCONST):   run;

NOTE: There were 1 observations read from the data set WORK._Q4.
NOTE: The data set WORK._Q4 has 1 observations and 9 variables.
NOTE: DATA statement used:
      real time             0.02 seconds
      cpu time              0.02 seconds


MPRINT(S2QCONST):   proc sort data=faconst(keep = ticker minwgt) out=_q5;
MPRINT(S2QCONST):   by ticker descending minwgt;

NOTE: There were 1 observations read from the data set WORK.FACONST.
NOTE: The data set WORK._Q5 has 1 observations and 2 variables.
NOTE: PROCEDURE SORT used:
      real time             0.02 seconds
      cpu time              0.00 seconds


MPRINT(S2QCONST):   data _q5;
MPRINT(S2QCONST):   set _q5;
MPRINT(S2QCONST):   by ticker descending minwgt;
MPRINT(S2QCONST):   if first.ticker;

NOTE: There were 1 observations read from the data set WORK._Q5.
NOTE: The data set WORK._Q5 has 1 observations and 2 variables.
NOTE: DATA statement used:
      real time             0.01 seconds
      cpu time              0.02 seconds


MPRINT(S2QCONST):   proc transpose data=_q5 out=_q6;
MPRINT(S2QCONST):   var minwgt;
MPRINT(S2QCONST):   id ticker;
NOTE: There were 1 observations read from the data set WORK._Q5.
NOTE: The data set WORK._Q6 has 1 observations and 2 variables.
NOTE: PROCEDURE TRANSPOSE used:
      real time             0.01 seconds
```

```
      cpu time            0.00 seconds


MPRINT(S2QCONST):   data _q6;
MPRINT(S2QCONST):   format _type_ $8. _name_ $8.;
MPRINT(S2QCONST):   set _q6;
MPRINT(S2QCONST):   _type_ = 'lowerbd ';
MPRINT(S2QCONST):   _name_ = 'LOWER';
MPRINT(S2QCONST):   _RHS_ = .;
MPRINT(S2QCONST):   if CPO eq . then CPO = 0;
MPRINT(S2QCONST):   if _BOND eq . then _BOND = 0;
MPRINT(S2QCONST):   if _CASH_ eq . then _CASH_ = 0;
MPRINT(S2QCONST):   if _EAFE eq . then _EAFE = 0;
MPRINT(S2QCONST):   if _ML_HYM eq . then _ML_HYM = 0;
MPRINT(S2QCONST):   if _W5000 eq . then _W5000 = 0;
MPRINT(S2QCONST):   run;

NOTE: There were 1 observations read from the data set WORK._Q6.
NOTE: The data set WORK._Q6 has 1 observations and 9 variables.
NOTE: DATA statement used:
      real time           0.02 seconds
      cpu time            0.01 seconds


MPRINT(S2QCONST):   data tsconst(keep = _type_ _name_ _rhs_ CPO _BOND _CASH_ _EAFE _ML_HYM _W5000);
MPRINT(S2QCONST):   set _q4 _q6;
MPRINT(S2QCONST):   run;

NOTE: There were 1 observations read from the data set WORK._Q4.
NOTE: There were 1 observations read from the data set WORK._Q6.
NOTE: The data set WORK.TSCONST has 2 observations and 9 variables.
NOTE: DATA statement used:
      real time           0.02 seconds
      cpu time            0.02 seconds


MPRINT(TRASH):   proc datasets nolist lib=work;
MPRINT(TRASH):   delete _q3 _q4 _q5 _q6;
MPRINT(TRASH):   run;

NOTE: Deleting WORK._Q3 (memtype=DATA).
NOTE: Deleting WORK._Q4 (memtype=DATA).
NOTE: Deleting WORK._Q5 (memtype=DATA).
NOTE: Deleting WORK._Q6 (memtype=DATA).
MPRINT(S2QCONST):  ;
MPRINT(QUAD):  ;
NOTE: PROCEDURE DATASETS used:
      real time           0.04 seconds
      cpu time            0.00 seconds


MPRINT(QUAD):   proc append base=pf.tarquad data=tsconst force;
MPRINT(QUAD):   run;

NOTE: Appending WORK.TSCONST to PF.TARQUAD.
NOTE: There were 2 observations read from the data set WORK.TSCONST.
NOTE: 2 observations added.
NOTE: The data set PF.TARQUAD has 9 observations and 9 variables.
```

```
NOTE: PROCEDURE APPEND used:
      real time               0.01 seconds
      cpu time                0.01 seconds


MPRINT(QUAD):   data _q1;
MPRINT(QUAD):   format ticker $8.;
MPRINT(QUAD):   ticker = "CPO";
MPRINT(QUAD):   wgt = 1 / 6;
MPRINT(QUAD):   output;
MPRINT(QUAD):   ticker = "_BOND";
MPRINT(QUAD):   wgt = 1 / 6;
MPRINT(QUAD):   output;
MPRINT(QUAD):   ticker = "_CASH_";
MPRINT(QUAD):   wgt = 1 / 6;
MPRINT(QUAD):   output;
MPRINT(QUAD):   ticker = "_EAFE";
MPRINT(QUAD):   wgt = 1 / 6;
MPRINT(QUAD):   output;
MPRINT(QUAD):   ticker = "_ML_HYM";
MPRINT(QUAD):   wgt = 1 / 6;
MPRINT(QUAD):   output;
MPRINT(QUAD):   ticker = "_W5000";
MPRINT(QUAD):   wgt = 1 / 6;
MPRINT(QUAD):   output;

NOTE: The data set WORK._Q1 has 6 observations and 2 variables.
NOTE: DATA statement used:
      real time               0.02 seconds
      cpu time                0.02 seconds


MPRINT(QUAD):   proc transpose data=_q1 out=_q1;
MPRINT(QUAD):   var wgt;
MPRINT(QUAD):   id ticker;
MPRINT(QUAD):   run;

NOTE: There were 6 observations read from the data set WORK._Q1.
NOTE: The data set WORK._Q1 has 1 observations and 7 variables.
NOTE: PROCEDURE TRANSPOSE used:
      real time               0.01 seconds
      cpu time                0.01 seconds


MPRINT(QUAD):   data _q1;
MPRINT(QUAD):   format _name_ $8.;
MPRINT(QUAD):   set _q1;
MPRINT(QUAD):   _type_ = 'PARMS   ';
MPRINT(QUAD):   _RHS_ = .;

NOTE: There were 1 observations read from the data set WORK._Q1.
NOTE: The data set WORK._Q1 has 1 observations and 9 variables.
NOTE: DATA statement used:
      real time               0.01 seconds
      cpu time                0.02 seconds


MPRINT(QUAD):   proc append base=pf.tarquad data=_q1 force;
```

```
MPRINT(QUAD):   run;

NOTE: Appending WORK._Q1 to PF.TARQUAD.
NOTE: There were 1 observations read from the data set WORK._Q1.
NOTE: 1 observations added.
NOTE: The data set PF.TARQUAD has 10 observations and 9 variables.
NOTE: PROCEDURE APPEND used:
      real time          0.01 seconds
      cpu time           0.00 seconds


MPRINT(T2QCONST):   proc sql noprint;
MPRINT(T2QCONST):   select count(*) into :cnt from aarange;
NOTE: PROCEDURE SQL used:
      real time          0.00 seconds
      cpu time           0.00 seconds


MPRINT(T2QCONST):   proc sql noprint;
MPRINT(T2QCONST):   select var, minwgt,maxwgt into :vname1 - :vname7, :minwgt1 - :minwgt7, :maxwgt1 - :maxwgt7 from aarange;
NOTE: PROCEDURE SQL used:
      real time          0.00 seconds
      cpu time           0.01 seconds


MPRINT(T2QCONST):   data _q7;
MPRINT(T2QCONST):   format _name_ $8. ticker $8. wgt 15.8 _type_ $8. _rhs_ 15.8;
MPRINT(T2QCONST):   if 1=2;
MPRINT(T2QCONST):   _name_='';
MPRINT(T2QCONST):   ticker = '';
MPRINT(T2QCONST):   wgt = .;
MPRINT(T2QCONST):   _type_ = '';
MPRINT(T2QCONST):   _rhs_ = .;
MPRINT(T2QCONST):   run;

NOTE: The data set WORK._Q7 has 0 observations and 5 variables.
NOTE: DATA statement used:
      real time          0.01 seconds
      cpu time           0.01 seconds


MPRINT(T2QCONST):   data _q7;
MPRINT(T2QCONST):   set _q7 quad_tv(in=inb keep=ticker STOCKS rename=(STOCKS=wgt ));
MPRINT(T2QCONST):   if inb then do;
MPRINT(T2QCONST):   _name_ = "STOCKS";
MPRINT(T2QCONST):   _type_ = 'ge';
MPRINT(T2QCONST):   _rhs_ = 0;
MPRINT(T2QCONST):   end;
MPRINT(T2QCONST):   run;

NOTE: There were 0 observations read from the data set WORK._Q7.
NOTE: There were 6 observations read from the data set WORK.QUAD_TV.
NOTE: The data set WORK._Q7 has 6 observations and 5 variables.
NOTE: DATA statement used:
      real time          0.02 seconds
      cpu time           0.01 seconds
```

```
MPRINT(T2QCONST):   data _q7;
MPRINT(T2QCONST):   set _q7 quad_tv(in=inb keep=ticker STOCKS rename=(STOCKS=wgt));
MPRINT(T2QCONST):   if inb then do;
MPRINT(T2QCONST):   _name_ = "STOCKS";
MPRINT(T2QCONST):   _type_ = 'le';
MPRINT(T2QCONST):   _rhs_ = 0.8925;
MPRINT(T2QCONST):   end;
MPRINT(T2QCONST):   run;

NOTE: There were 6 observations read from the data set WORK._Q7.
NOTE: There were 6 observations read from the data set WORK.QUAD_TV.
NOTE: The data set WORK._Q7 has 12 observations and 5 variables.
NOTE: DATA statement used:
      real time          0.02 seconds
      cpu time           0.02 seconds


MPRINT(T2QCONST):   data _q7;
MPRINT(T2QCONST):   set _q7 quad_tv(in=inb keep=ticker BONDS rename=(BONDS=wgt ));
MPRINT(T2QCONST):   if inb then do;
MPRINT(T2QCONST):   _name_ = "BONDS";
MPRINT(T2QCONST):   _type_ = 'ge';
MPRINT(T2QCONST):   _rhs_ = 0;
MPRINT(T2QCONST):   end;
MPRINT(T2QCONST):   run;

NOTE: There were 12 observations read from the data set WORK._Q7.
NOTE: There were 6 observations read from the data set WORK.QUAD_TV.
NOTE: The data set WORK._Q7 has 18 observations and 5 variables.
NOTE: DATA statement used:
      real time          0.02 seconds
      cpu time           0.01 seconds


MPRINT(T2QCONST):   data _q7;
MPRINT(T2QCONST):   set _q7 quad_tv(in=inb keep=ticker BONDS rename=(BONDS=wgt));
MPRINT(T2QCONST):   if inb then do;
MPRINT(T2QCONST):   _name_ = "BONDS";
MPRINT(T2QCONST):   _type_ = 'le';
MPRINT(T2QCONST):   _rhs_ = 1;
MPRINT(T2QCONST):   end;
MPRINT(T2QCONST):   run;

NOTE: There were 18 observations read from the data set WORK._Q7.
NOTE: There were 6 observations read from the data set WORK.QUAD_TV.
NOTE: The data set WORK._Q7 has 24 observations and 5 variables.
NOTE: DATA statement used:
      real time          0.01 seconds
      cpu time           0.01 seconds


MPRINT(T2QCONST):   data _q7;
MPRINT(T2QCONST):   set _q7 quad_tv(in=inb keep=ticker IGBONDS rename=(IGBONDS=wgt ));
MPRINT(T2QCONST):   if inb then do;
MPRINT(T2QCONST):   _name_ = "IGBONDS";
MPRINT(T2QCONST):   _type_ = 'ge';
MPRINT(T2QCONST):   _rhs_ = 0.1125;
MPRINT(T2QCONST):   end;
```

```
MPRINT(T2QCONST):   run;

NOTE: There were 24 observations read from the data set WORK._Q7.
NOTE: There were 6 observations read from the data set WORK.QUAD_TV.
NOTE: The data set WORK._Q7 has 30 observations and 5 variables.
NOTE: DATA statement used:
      real time           0.02 seconds
      cpu time            0.02 seconds


MPRINT(T2QCONST):   data _q7;
MPRINT(T2QCONST):   set _q7 quad_tv(in=inb keep=ticker IGBONDS rename=(IGBONDS=wgt));
MPRINT(T2QCONST):   if inb then do;
MPRINT(T2QCONST):   _name_ = "IGBONDS";
MPRINT(T2QCONST):   _type_ = 'le';
MPRINT(T2QCONST):   _rhs_ = 1;
MPRINT(T2QCONST):   end;
MPRINT(T2QCONST):   run;

NOTE: There were 30 observations read from the data set WORK._Q7.
NOTE: There were 6 observations read from the data set WORK.QUAD_TV.
NOTE: The data set WORK._Q7 has 36 observations and 5 variables.
NOTE: DATA statement used:
      real time           0.01 seconds
      cpu time            0.02 seconds


MPRINT(T2QCONST):   data _q7;
MPRINT(T2QCONST):   set _q7 quad_tv(in=inb keep=ticker CASH rename=(CASH=wgt ));
MPRINT(T2QCONST):   if inb then do;
MPRINT(T2QCONST):   _name_ = "CASH";
MPRINT(T2QCONST):   _type_ = 'ge';
MPRINT(T2QCONST):   _rhs_ = 0;
MPRINT(T2QCONST):   end;
MPRINT(T2QCONST):   run;

NOTE: There were 36 observations read from the data set WORK._Q7.
NOTE: There were 6 observations read from the data set WORK.QUAD_TV.
NOTE: The data set WORK._Q7 has 42 observations and 5 variables.
NOTE: DATA statement used:
      real time           0.01 seconds
      cpu time            0.02 seconds


MPRINT(T2QCONST):   data _q7;
MPRINT(T2QCONST):   set _q7 quad_tv(in=inb keep=ticker CASH rename=(CASH=wgt));
MPRINT(T2QCONST):   if inb then do;
MPRINT(T2QCONST):   _name_ = "CASH";
MPRINT(T2QCONST):   _type_ = 'le';
MPRINT(T2QCONST):   _rhs_ = 1;
MPRINT(T2QCONST):   end;
MPRINT(T2QCONST):   run;

NOTE: There were 42 observations read from the data set WORK._Q7.
NOTE: There were 6 observations read from the data set WORK.QUAD_TV.
NOTE: The data set WORK._Q7 has 48 observations and 5 variables.
NOTE: DATA statement used:
      real time           0.01 seconds
```

```
      cpu time            0.02 seconds


MPRINT(T2QCONST):   data _q7;
MPRINT(T2QCONST):   set _q7 quad_tv(in=inb keep=ticker FSTOCKS rename=(FSTOCKS=wgt ));
MPRINT(T2QCONST):   if inb then do;
MPRINT(T2QCONST):   _name_ = "FSTOCKS";
MPRINT(T2QCONST):   _type_ = 'ge';
MPRINT(T2QCONST):   _rhs_ = 0;
MPRINT(T2QCONST):   end;
MPRINT(T2QCONST):   run;

NOTE: There were 48 observations read from the data set WORK._Q7.
NOTE: There were 6 observations read from the data set WORK.QUAD_TV.
NOTE: The data set WORK._Q7 has 54 observations and 5 variables.
NOTE: DATA statement used:
      real time           0.02 seconds
      cpu time            0.01 seconds


MPRINT(T2QCONST):   data _q7;
MPRINT(T2QCONST):   set _q7 quad_tv(in=inb keep=ticker FSTOCKS rename=(FSTOCKS=wgt));
MPRINT(T2QCONST):   if inb then do;
MPRINT(T2QCONST):   _name_ = "FSTOCKS";
MPRINT(T2QCONST):   _type_ = 'le';
MPRINT(T2QCONST):   _rhs_ = 0.18;
MPRINT(T2QCONST):   end;
MPRINT(T2QCONST):   run;

NOTE: There were 54 observations read from the data set WORK._Q7.
NOTE: There were 6 observations read from the data set WORK.QUAD_TV.
NOTE: The data set WORK._Q7 has 60 observations and 5 variables.
NOTE: DATA statement used:
      real time           0.01 seconds
      cpu time            0.01 seconds


MPRINT(T2QCONST):   data _q7;
MPRINT(T2QCONST):   set _q7 quad_tv(in=inb keep=ticker HYBONDS rename=(HYBONDS=wgt ));
MPRINT(T2QCONST):   if inb then do;
MPRINT(T2QCONST):   _name_ = "HYBONDS";
MPRINT(T2QCONST):   _type_ = 'ge';
MPRINT(T2QCONST):   _rhs_ = 0;
MPRINT(T2QCONST):   end;
MPRINT(T2QCONST):   run;

NOTE: There were 60 observations read from the data set WORK._Q7.
NOTE: There were 6 observations read from the data set WORK.QUAD_TV.
NOTE: The data set WORK._Q7 has 66 observations and 5 variables.
NOTE: DATA statement used:
      real time           0.02 seconds
      cpu time            0.01 seconds


MPRINT(T2QCONST):   data _q7;
MPRINT(T2QCONST):   set _q7 quad_tv(in=inb keep=ticker HYBONDS rename=(HYBONDS=wgt));
MPRINT(T2QCONST):   if inb then do;
MPRINT(T2QCONST):   _name_ = "HYBONDS";
```

```
MPRINT(T2QCONST):   _type_ = 'le';
MPRINT(T2QCONST):   _rhs_ = 0.1;
MPRINT(T2QCONST):   end;
MPRINT(T2QCONST):   run;

NOTE: There were 66 observations read from the data set WORK._Q7.
NOTE: There were 6 observations read from the data set WORK.QUAD_TV.
NOTE: The data set WORK._Q7 has 72 observations and 5 variables.
NOTE: DATA statement used:
      real time          0.02 seconds
      cpu time           0.02 seconds


MPRINT(T2QCONST):   data _q7;
MPRINT(T2QCONST):   set _q7 quad_tv(in=inb keep=ticker DSTOCKS rename=(DSTOCKS=wgt ));
MPRINT(T2QCONST):   if inb then do;
MPRINT(T2QCONST):   _name_ = "DSTOCKS";
MPRINT(T2QCONST):   _type_ = 'ge';
MPRINT(T2QCONST):   _rhs_ = 0;
MPRINT(T2QCONST):   end;
MPRINT(T2QCONST):   run;

NOTE: There were 72 observations read from the data set WORK._Q7.
NOTE: There were 6 observations read from the data set WORK.QUAD_TV.
NOTE: The data set WORK._Q7 has 78 observations and 5 variables.
NOTE: DATA statement used:
      real time          0.02 seconds
      cpu time           0.01 seconds


MPRINT(T2QCONST):   data _q7;
MPRINT(T2QCONST):   set _q7 quad_tv(in=inb keep=ticker DSTOCKS rename=(DSTOCKS=wgt));
MPRINT(T2QCONST):   if inb then do;
MPRINT(T2QCONST):   _name_ = "DSTOCKS";
MPRINT(T2QCONST):   _type_ = 'le';
MPRINT(T2QCONST):   _rhs_ = 0.94;
MPRINT(T2QCONST):   end;
MPRINT(T2QCONST):   run;

NOTE: There were 78 observations read from the data set WORK._Q7.
NOTE: There were 6 observations read from the data set WORK.QUAD_TV.
NOTE: The data set WORK._Q7 has 84 observations and 5 variables.
NOTE: DATA statement used:
      real time          0.02 seconds
      cpu time           0.02 seconds


MPRINT(T2QCONST):   proc sort data=_q7;
MPRINT(T2QCONST):   by _name_ _type_ _rhs_;
MPRINT(T2QCONST):   run;

NOTE: There were 84 observations read from the data set WORK._Q7.
NOTE: The data set WORK._Q7 has 84 observations and 5 variables.
NOTE: PROCEDURE SORT used:
      real time          0.01 seconds
      cpu time           0.02 seconds
```

```
MPRINT(T2QCONST):   proc transpose data=_q7 out=ttconst(drop=_label_);
MPRINT(T2QCONST):   by _name_ _type_ _rhs_;
MPRINT(T2QCONST):   id ticker;
MPRINT(T2QCONST):   var wgt;
MPRINT(T2QCONST):   run;

NOTE: There were 84 observations read from the data set WORK._Q7.
NOTE: The data set WORK.TTCONST has 14 observations and 9 variables.
NOTE: PROCEDURE TRANSPOSE used:
      real time             0.02 seconds
      cpu time              0.02 seconds


MPRINT(TRASH):   proc datasets nolist lib=work;
MPRINT(TRASH):   delete _q7;
MPRINT(TRASH):   run;

NOTE: Deleting WORK._Q7 (memtype=DATA).
MPRINT(T2QCONST):  ;
MPRINT(QUAD):  ;
NOTE: PROCEDURE DATASETS used:
      real time             0.02 seconds
      cpu time              0.00 seconds


MPRINT(QUAD):   proc append base=pf.tarquad data=ttconst force;
MPRINT(QUAD):   run;

NOTE: Appending WORK.TTCONST to PF.TARQUAD.
NOTE: There were 14 observations read from the data set WORK.TTCONST.
NOTE: 14 observations added.
NOTE: The data set PF.TARQUAD has 24 observations and 9 variables.
NOTE: PROCEDURE APPEND used:
      real time             0.01 seconds
      cpu time              0.00 seconds


MPRINT(QUAD):   data _null_;
MPRINT(QUAD):   title "Minimum Tracking Error Portfolio";
MPRINT(QUAD):   run;

NOTE: DATA statement used:
      real time             0.00 seconds
      cpu time              0.00 seconds


MPRINT(CALLNLP):   proc nlp inquad=pf.tarquad outest=pf.taroutquad cov=2 ;
MPRINT(CALLNLP):   min;
MPRINT(CALLNLP):   parms CPO _BOND _CASH_ _EAFE _ML_HYM _W5000 ;
MPRINT(CALLNLP):   run;

NOTE: Your code contains 0 program statements.
NOTE: Analytic derivatives are used with a quadratic optimization problem.
NOTE: The Hessian matrix of the QP is singular.
NOTE: Initial point was changed to be feasible for boundary and linear constraints.
NOTE: ABSGCONV convergence criterion satisfied.
NOTE: There were 24 observations read from the data set PF.TARQUAD.
NOTE: The data set PF.TAROUTQUAD has 47 observations and 11 variables.
```

```
NOTE: The PROCEDURE NLP printed pages 1-4.
NOTE: PROCEDURE NLP used:
      real time           0.50 seconds
      cpu time            0.06 seconds


MPRINT(CALLNLP):   proc sql noprint;
MPRINT(CALLNLP):   select _rhs_ into :objfnval from pf.taroutquad where _type_ = 'PARMS';
MPRINT(QUAD):  ;
NOTE: PROCEDURE SQL used:
      real time           0.00 seconds
      cpu time            0.01 seconds


MPRINT(TRASH):   proc datasets nolist lib=work;
MPRINT(TRASH):   delete _q1 _q2;
MPRINT(TRASH):   run;

NOTE: The file WORK._Q2 (memtype=DATA) was not found, but appears on a DELETE statement.
NOTE: Deleting WORK._Q1 (memtype=DATA).
MPRINT(QUAD):  ;
MPRINT(TARADJ):  ;

NOTE: PROCEDURE DATASETS used:
      real time           0.01 seconds
      cpu time            0.01 seconds


MPRINT(Q2CTW):   proc sql;
MPRINT(Q2CTW):   select _rhs_ into :objfnval from pf.taroutquad where _type_ = 'PARMS';
NOTE: The PROCEDURE SQL printed page 5.
NOTE: PROCEDURE SQL used:
      real time           0.00 seconds
      cpu time            0.01 seconds


MPRINT(Q2CTW):   data newtar(drop=_type_ _rhs_);
MPRINT(Q2CTW):   format cticker $8.;
MPRINT(Q2CTW):   set pf.taroutquad(drop= _name_ _iter_ _tech_ where=(_type_='PARMS'));
MPRINT(Q2CTW):   cticker = "_newtar";
MPRINT(Q2CTW):   call symput("OBJFNVAL",put(_rhs_,best.));
MPRINT(Q2CTW):   if _rhs_ ne . then do;
MPRINT(Q2CTW):   output;
MPRINT(Q2CTW):   call symput("OSTATUS","0");
MPRINT(Q2CTW):   end;
MPRINT(Q2CTW):   else do;
MPRINT(Q2CTW):   call symput("OSTATUS","1");
MPRINT(Q2CTW):   end;
MPRINT(Q2CTW):   run;

NOTE: There were 1 observations read from the data set PF.TAROUTQUAD.
      WHERE _type_='PARMS';
NOTE: The data set WORK.NEWTAR has 1 observations and 7 variables.
NOTE: DATA statement used:
      real time           0.01 seconds
      cpu time            0.01 seconds
```

```
MPRINT(Q2CTW):   proc transpose data=newtar out=newtar(rename=(_name_=ticker col1=wgt));
MPRINT(Q2CTW):   by cticker;
MPRINT(Q2CTW):   run;

NOTE: There were 1 observations read from the data set WORK.NEWTAR.
NOTE: The data set WORK.NEWTAR has 6 observations and 3 variables.
NOTE: PROCEDURE TRANSPOSE used:
      real time          0.02 seconds
      cpu time           0.02 seconds


MPRINT(TARADJ):  ;
MPRINT(CTWRET):   data _r7;
MPRINT(CTWRET):   set newtar(where=(wgt ne 0));
MPRINT(CTWRET):   run;

NOTE: There were 5 observations read from the data set WORK.NEWTAR.
      WHERE wgt not = 0;
NOTE: The data set WORK._R7 has 5 observations and 3 variables.
NOTE: DATA statement used:
      real time          0.01 seconds
      cpu time           0.01 seconds


MPRINT(VARENUM):   proc sql;
MPRINT(VARENUM):   create table _r2 as select distinct ticker as ticker from _r7 order by ticker;
NOTE: Table WORK._R2 created, with 5 rows and 1 columns.

NOTE: PROCEDURE SQL used:
      real time          0.01 seconds
      cpu time           0.00 seconds


MPRINT(VARENUM):   data _r2;
MPRINT(VARENUM):   set _r2;
MPRINT(VARENUM):   format tickid $8.;
MPRINT(VARENUM):   tickid = "_W" || left(trim(put(_n_,8.)));
MPRINT(VARENUM):   run;

NOTE: There were 5 observations read from the data set WORK._R2.
NOTE: The data set WORK._R2 has 5 observations and 2 variables.
NOTE: DATA statement used:
      real time          0.01 seconds
      cpu time           0.00 seconds


MPRINT(CTWRET):  ;
MPRINT(DS2MA):   proc sql noprint;
MPRINT(DS2MA):   select count(distinct ticker) into :_ct0 from _r2 where ticker ne '' ;
NOTE: PROCEDURE SQL used:
      real time          0.01 seconds
      cpu time           0.01 seconds


MPRINT(DS2MA):   proc sql noprint;
MPRINT(DS2MA):   select distinct ticker into :_ct1 - :_ct5 from _r2 where ticker ne '' order by ticker;
MPRINT(CTWRET):  ;
NOTE: PROCEDURE SQL used:
```

```
      real time           0.00 seconds
      cpu time            0.01 seconds


MPRINT(V_REPLAC):   proc sort data=_r7 out=_a1;
MPRINT(V_REPLAC):   by ticker;

NOTE: There were 5 observations read from the data set WORK._R7.
NOTE: The data set WORK._A1 has 5 observations and 3 variables.
NOTE: PROCEDURE SORT used:
      real time           0.01 seconds
      cpu time            0.01 seconds


MPRINT(V_REPLAC):   proc sort data=_r2(keep=ticker tickid) out=_a2;
MPRINT(V_REPLAC):   by ticker;
NOTE: There were 5 observations read from the data set WORK._R2.
NOTE: The data set WORK._A2 has 5 observations and 2 variables.
NOTE: PROCEDURE SORT used:
      real time           0.01 seconds
      cpu time            0.00 seconds


MPRINT(V_REPLAC):   data _r3 (drop=ticker rename=(__x=ticker));
MPRINT(V_REPLAC):   merge _a1(in=ina) _a2(rename=(tickid = __x ticker=ticker));
MPRINT(V_REPLAC):   by ticker;
MPRINT(V_REPLAC):   if ina;
MPRINT(V_REPLAC):   run;

NOTE: There were 5 observations read from the data set WORK._A1.
NOTE: There were 5 observations read from the data set WORK._A2.
NOTE: The data set WORK._R3 has 5 observations and 3 variables.
NOTE: DATA statement used:
      real time           0.02 seconds
      cpu time            0.03 seconds


MPRINT(TRASH):   proc datasets nolist lib=work;
MPRINT(TRASH):   delete _a1 _a2;
MPRINT(TRASH):   run;

NOTE: Deleting WORK._A1 (memtype=DATA).
NOTE: Deleting WORK._A2 (memtype=DATA).
MPRINT(V_REPLAC):  ;
MPRINT(CTWRET):  ;

NOTE: PROCEDURE DATASETS used:
      real time           0.02 seconds
      cpu time            0.00 seconds


MPRINT(L2WTRAN):   proc sort data=_r3;
MPRINT(L2WTRAN):   by cticker;
MPRINT(L2WTRAN):   run;

NOTE: There were 5 observations read from the data set WORK._R3.
NOTE: The data set WORK._R3 has 5 observations and 3 variables.
NOTE: PROCEDURE SORT used:
```

```
      real time           0.01 seconds
      cpu time            0.00 seconds


MPRINT(L2WTRAN):   proc transpose data=_r3(keep=wgt ticker cticker) out=_r3(keep=cticker _numeric_);
MPRINT(L2WTRAN):   var wgt;
MPRINT(L2WTRAN):   id ticker;
MPRINT(L2WTRAN):   by cticker;
MPRINT(L2WTRAN):   run;

NOTE: There were 5 observations read from the data set WORK._R3.
NOTE: The data set WORK._R3 has 1 observations and 6 variables.
NOTE: PROCEDURE TRANSPOSE used:
      real time           0.02 seconds
      cpu time            0.01 seconds


MPRINT(CTWRET):  ;
MPRINT(DSNONULL):   data _r3;
MPRINT(DSNONULL):   set _r3;
MPRINT(DSNONULL):   array _x _numeric_;
MPRINT(DSNONULL):   do over _x;
MPRINT(DSNONULL):   if _x = . then _x = 0;
MPRINT(DSNONULL):   end;
MPRINT(DSNONULL):   run;

NOTE: There were 1 observations read from the data set WORK._R3.
NOTE: The data set WORK._R3 has 1 observations and 6 variables.
NOTE: DATA statement used:
      real time           0.01 seconds
      cpu time            0.01 seconds


MPRINT(CTWRET):  ;
MPRINT(CLEANRET):   proc sql;
MPRINT(CLEANRET):   create table _rc2 as select distinct ticker from _r2;
NOTE: Table WORK._RC2 created, with 5 rows and 1 columns.

NOTE: PROCEDURE SQL used:
      real time           0.01 seconds
      cpu time            0.00 seconds


MPRINT(CLEANRET):   data _rc2;
MPRINT(CLEANRET):   set _rc2 end=eof;
MPRINT(CLEANRET):   if _n_ eq 1 then call symput("_CRA",ticker);
MPRINT(CLEANRET):   if eof then call symput("_CRZ",ticker);
MPRINT(CLEANRET):   return = 0;
MPRINT(CLEANRET):   run;

NOTE: There were 5 observations read from the data set WORK._RC2.
NOTE: The data set WORK._RC2 has 5 observations and 2 variables.
NOTE: DATA statement used:
      real time           0.01 seconds
      cpu time            0.00 seconds


MPRINT(L2WTRAN):   proc transpose data=_rc2(keep=return ticker ) out=_rc2(keep= _numeric_);
```

```
MPRINT(L2WTRAN):   var return;
MPRINT(L2WTRAN):   id ticker;
MPRINT(L2WTRAN):  ;
MPRINT(L2WTRAN):   run;

NOTE: There were 5 observations read from the data set WORK._RC2.
NOTE: The data set WORK._RC2 has 1 observations and 5 variables.
NOTE: PROCEDURE TRANSPOSE used:
      real time             0.01 seconds
      cpu time              0.00 seconds


MPRINT(CLEANRET):  ;
MPRINT(CLEANRET):   data _r5(keep=date CPO -- _W5000 );
MPRINT(CLEANRET):   merge _rc2 _tadtt;
MPRINT(CLEANRET):   output;
MPRINT(CLEANRET):   run;

NOTE: There were 1 observations read from the data set WORK._RC2.
NOTE: There were 36 observations read from the data set WORK._TADTT.
NOTE: The data set WORK._R5 has 36 observations and 6 variables.
NOTE: DATA statement used:
      real time             0.02 seconds
      cpu time              0.00 seconds


MPRINT(TRASH):   proc datasets nolist lib=work;
MPRINT(TRASH):   delete _rc2;
MPRINT(TRASH):   run;

NOTE: Deleting WORK._RC2 (memtype=DATA).
MPRINT(CLEANRET):  ;
MPRINT(CTWRET):  ;

NOTE: PROCEDURE DATASETS used:
      real time             0.02 seconds
      cpu time              0.01 seconds


MPRINT(CTWRET):   data _r6;
MPRINT(CTWRET):   set _r5(rename=( CPO=_R1 _BOND=_R2 _CASH_=_R3 _EAFE=_R4 _W5000=_R5 ));
MPRINT(CTWRET):   run;

NOTE: There were 36 observations read from the data set WORK._R5.
NOTE: The data set WORK._R6 has 36 observations and 6 variables.
NOTE: DATA statement used:
      real time             0.01 seconds
      cpu time              0.00 seconds


MPRINT(CTWRET):   data _r7(keep=cticker date return);
MPRINT(CTWRET):   set _r3 nobs=_nobs;
MPRINT(CTWRET):   array _w[5] _w1 - _w5;
MPRINT(CTWRET):   array _nw[5];
MPRINT(CTWRET):   retain _nw1 - _nw5;
MPRINT(CTWRET):   retain _w1 - _w5 ;
MPRINT(CTWRET):   do _i = 1 to 5;
MPRINT(CTWRET):   _nw[_i] = _w[_i];
```

```
MPRINT(CTWRET):   end;
MPRINT(CTWRET):   _zportf + 1;
MPRINT(CTWRET):   if _nobs > 50 and 10*int(_zportf/10) = _zportf then put "portfolio " _zportf " / " _nobs;
MPRINT(CTWRET):   do _rno = 1 to _robs;
MPRINT(CTWRET):   set _r6 nobs=_robs point=_rno;
MPRINT(CTWRET):   array _r[5] _r1 - _r5;
MPRINT(CTWRET):   _sum = 0;
MPRINT(CTWRET):   return = 0;
MPRINT(CTWRET):   do _i = 1 to 5;
MPRINT(CTWRET):   if _nw[_i] ne 0 then do;
MPRINT(CTWRET):   return = return + _nw[_i] * _r[_i];
MPRINT(CTWRET):   end;
MPRINT(CTWRET):   end;
MPRINT(CTWRET):   output _r7;
MPRINT(CTWRET):  ;
MPRINT(CTWRET):   end;
MPRINT(CTWRET):  ;
MPRINT(CTWRET):   run;

NOTE: There were 1 observations read from the data set WORK._R3.
NOTE: The data set WORK._R7 has 36 observations and 3 variables.
NOTE: DATA statement used:
      real time           0.02 seconds
      cpu time            0.01 seconds


MPRINT(CTWRET):   proc sort data=_r7;
MPRINT(CTWRET):   by date cticker;

NOTE: There were 36 observations read from the data set WORK._R7.
NOTE: The data set WORK._R7 has 36 observations and 3 variables.
NOTE: PROCEDURE SORT used:
      real time           0.01 seconds
      cpu time            0.00 seconds


MPRINT(CTWRET):   proc transpose data=_r7 out=_dtt(keep = _numeric_);
MPRINT(CTWRET):   var return;
MPRINT(CTWRET):   id cticker;
MPRINT(CTWRET):   by date;
MPRINT(CTWRET):   run;

NOTE: There were 36 observations read from the data set WORK._R7.
NOTE: The data set WORK._DTT has 36 observations and 2 variables.
NOTE: PROCEDURE TRANSPOSE used:
      real time           0.02 seconds
      cpu time            0.01 seconds


MPRINT(TERR_CTW):  ;
MPRINT(W2LTRAN):   proc transpose data=_dtt out=_dtr(keep = date ticker col1 rename=(col1=return)) name=ticker;
MPRINT(W2LTRAN):   by date;
MPRINT(W2LTRAN):   run;

NOTE: There were 36 observations read from the data set WORK._DTT.
NOTE: The data set WORK._DTR has 36 observations and 3 variables.
NOTE: PROCEDURE TRANSPOSE used:
      real time           0.02 seconds
```

```
      cpu time            0.00 seconds

MPRINT(W2LTRAN):   data _dtr;
MPRINT(W2LTRAN):   length ticker $8;
MPRINT(W2LTRAN):   set _dtr(where=(return ne .));
MPRINT(W2LTRAN):   run;

NOTE: There were 36 observations read from the data set WORK._DTR.
      WHERE return not = .;
NOTE: The data set WORK._DTR has 36 observations and 3 variables.
NOTE: DATA statement used:
      real time           0.01 seconds
      cpu time            0.01 seconds


MPRINT(DTT2DTR):  ;
MPRINT(TERR_CTW):  ;
MPRINT(TERR_CTW):   proc sql noprint;
MPRINT(TERR_CTW):   select std(return-0)*sqrt(12) into :tarterr from _dtr a, _tadtt b where a.date=b.date;
tracking error = 0.147216
MPRINT(TARADJ):  ;
Adjusted Target Tracking Error = 0.147216
NOTE: PROCEDURE SQL used:
      real time           0.02 seconds
      cpu time            0.01 seconds


MPRINT(TARADJ):   proc sql noprint;
MPRINT(TARADJ):   create table _newtar as select sum(wgt*(dstocks+fstocks)) as stocks, sum(wgt*fstocks) as fstocks,
sum(wgt*(igbonds+hybonds)) as bonds, sum(wgt*cash) as cash from quad_tv a, newtar b where a.ticker=b.ticker;
NOTE: Table WORK._NEWTAR created, with 1 rows and 4 columns.

NOTE: PROCEDURE SQL used:
      real time           0.04 seconds
      cpu time            0.01 seconds


MPRINT(W2LTRAN):   proc transpose data=_newtar out=_newtar(keep = var col1 rename=(col1=newtar)) name=var;
MPRINT(W2LTRAN):  ;
MPRINT(W2LTRAN):   run;

NOTE: There were 1 observations read from the data set WORK._NEWTAR.
NOTE: The data set WORK._NEWTAR has 4 observations and 2 variables.
NOTE: PROCEDURE TRANSPOSE used:
      real time           0.01 seconds
      cpu time            0.02 seconds


MPRINT(W2LTRAN):   data _newtar;
MPRINT(W2LTRAN):   length var $8;
MPRINT(W2LTRAN):   set _newtar(where=(newtar ne .));
MPRINT(W2LTRAN):   run;

NOTE: There were 4 observations read from the data set WORK._NEWTAR.
      WHERE newtar not = .;
NOTE: The data set WORK._NEWTAR has 4 observations and 2 variables.
NOTE: DATA statement used:
```

```
      real time           0.01 seconds
      cpu time            0.01 seconds


MPRINT(TARADJ):  ;
MPRINT(TARADJ):   proc sort data=_newtar;
MPRINT(TARADJ):   by var;
MPRINT(TARADJ):   run;

NOTE: There were 4 observations read from the data set WORK._NEWTAR.
NOTE: The data set WORK._NEWTAR has 4 observations and 2 variables.
NOTE: PROCEDURE SORT used:
      real time           0.01 seconds
      cpu time            0.00 seconds


MPRINT(TARADJ):   proc sort data=pf.aaconst;
MPRINT(TARADJ):   by var;
NOTE: There were 6 observations read from the data set PF.AACONST.
NOTE: The data set PF.AACONST has 6 observations and 6 variables.
NOTE: PROCEDURE SORT used:
      real time           0.02 seconds
      cpu time            0.02 seconds


MPRINT(TARADJ):   data pf.adjconst(drop=newtar);
MPRINT(TARADJ):   merge _newtar(in=inb) pf.aaconst;
MPRINT(TARADJ):   by var;
MPRINT(TARADJ):   if newtar ne . then do;
MPRINT(TARADJ):   target=newtar;
MPRINT(TARADJ):   end;
MPRINT(TARADJ):   minwgt=target-range;
MPRINT(TARADJ):   maxwgt=target+range;
MPRINT(TARADJ):   run;

NOTE: There were 4 observations read from the data set WORK._NEWTAR.
NOTE: There were 6 observations read from the data set PF.AACONST.
NOTE: The data set PF.ADJCONST has 6 observations and 6 variables.
NOTE: DATA statement used:
      real time           0.02 seconds
      cpu time            0.01 seconds


MPRINT(OPTO_PF):  ;
MPRINT(OPTO_PF):  ;
MPRINT(DS2MA):   proc sql noprint;
MPRINT(DS2MA):   select count(distinct ticker) into :_ma0 from pf.initds where ticker ne '' ;
NOTE: PROCEDURE SQL used:
      real time           0.02 seconds
      cpu time            0.01 seconds


MPRINT(DS2MA):   proc sql noprint;
MPRINT(DS2MA):   select distinct ticker into :_ma1 - :_ma9 from pf.initds where ticker ne '' order by ticker;
MPRINT(OPTO_CTW):  ;
NOTE: PROCEDURE SQL used:
      real time           0.00 seconds
      cpu time            0.01 seconds
```

```
MPRINT(DS2MA):   proc sql noprint;
MPRINT(DS2MA):   select count(distinct cticker) into :_p0 from pf.initds where cticker ne '' ;
NOTE: PROCEDURE SQL used:
      real time           0.01 seconds
      cpu time            0.01 seconds


MPRINT(DS2MA):   proc sql noprint;
MPRINT(DS2MA):   select distinct cticker into :_p1 - :_p1 from pf.initds where cticker ne '' order by cticker;
MPRINT(OPTO_CTW):  ;
NOTE: PROCEDURE SQL used:
      real time           0.00 seconds
      cpu time            0.01 seconds


MPRINT(OPTO_CTW):   proc sql noprint;
MPRINT(OPTO_CTW):   create table _ticks as select distinct ticker from pf.initds;
NOTE: Table WORK._TICKS created, with 9 rows and 1 columns.

MPRINT(OPTO_CTW):   insert into _ticks(ticker) values ("_BENCH");
NOTE: 1 row was inserted into WORK._TICKS.

NOTE: PROCEDURE SQL used:
      real time           0.02 seconds
      cpu time            0.00 seconds


MPRINT(OPTO_CTW):   proc sort data=_ticks;
MPRINT(OPTO_CTW):   by ticker;
MPRINT(OPTO_CTW):   run;

NOTE: There were 10 observations read from the data set WORK._TICKS.
NOTE: The data set WORK._TICKS has 10 observations and 1 variables.
NOTE: PROCEDURE SORT used:
      real time           0.01 seconds
      cpu time            0.01 seconds


MPRINT(CLEANRET):   proc sql;
MPRINT(CLEANRET):   create table _rc2 as select distinct ticker from _ticks;
NOTE: Table WORK._RC2 created, with 10 rows and 1 columns.

NOTE: PROCEDURE SQL used:
      real time           0.01 seconds
      cpu time            0.02 seconds


MPRINT(CLEANRET):   data _rc2;
MPRINT(CLEANRET):   set _rc2 end=eof;
MPRINT(CLEANRET):   if _n_ eq 1 then call symput("_CRA",ticker);
MPRINT(CLEANRET):   if eof then call symput("_CRZ",ticker);
MPRINT(CLEANRET):   return = 0;
MPRINT(CLEANRET):   run;

NOTE: There were 10 observations read from the data set WORK._RC2.
NOTE: The data set WORK._RC2 has 10 observations and 2 variables.
```

```
NOTE: DATA statement used:
      real time           0.01 seconds
      cpu time            0.02 seconds


MPRINT(L2WTRAN):   proc transpose data=_rc2(keep=return ticker ) out=_rc2(keep= _numeric_);
MPRINT(L2WTRAN):   var return;
MPRINT(L2WTRAN):   id ticker;
MPRINT(L2WTRAN):  ;
MPRINT(L2WTRAN):   run;

NOTE: There were 10 observations read from the data set WORK._RC2.
NOTE: The data set WORK._RC2 has 1 observations and 10 variables.
NOTE: PROCEDURE TRANSPOSE used:
      real time           0.01 seconds
      cpu time            0.02 seconds


MPRINT(CLEANRET):  ;
MPRINT(CLEANRET):   data _dtt2(keep=date CPO -- _GIC_ );
MPRINT(CLEANRET):   merge _rc2 pf.mdl_dtt;
MPRINT(CLEANRET):   array __x CPO -- _GIC_ ;
MPRINT(CLEANRET):   do over __x;
MPRINT(CLEANRET):   if __x = . then __x = 0;
MPRINT(CLEANRET):   end;
MPRINT(CLEANRET):   output;
MPRINT(CLEANRET):   run;

NOTE: There were 1 observations read from the data set WORK._RC2.
NOTE: There were 36 observations read from the data set PF.MDL_DTT.
NOTE: The data set WORK._DTT2 has 36 observations and 11 variables.
NOTE: DATA statement used:
      real time           0.03 seconds
      cpu time            0.01 seconds


MPRINT(TRASH):   proc datasets nolist lib=work;
MPRINT(TRASH):   delete _rc2;
MPRINT(TRASH):   run;

NOTE: Deleting WORK._RC2 (memtype=DATA).
MPRINT(CLEANRET):  ;
MPRINT(OPTO_CTW):  ;

NOTE: PROCEDURE DATASETS used:
      real time           0.02 seconds
      cpu time            0.01 seconds


MPRINT(OPTO_CTW):   data _dtt1(drop=_name);
MPRINT(OPTO_CTW):   set _dtt2;
MPRINT(OPTO_CTW):   format _name $8.;
MPRINT(OPTO_CTW):   array __x _numeric_;
MPRINT(OPTO_CTW):   do over __x;
MPRINT(OPTO_CTW):   call vname(__x,_name);
MPRINT(OPTO_CTW):   if _name ne "_BENCH" and _name ne "DATE" then __x = __x - _BENCH;
MPRINT(OPTO_CTW):   end;
```

```
NOTE: There were 36 observations read from the data set WORK._DTT2.
NOTE: The data set WORK._DTT1 has 36 observations and 11 variables.
NOTE: DATA statement used:
      real time           0.02 seconds
      cpu time            0.02 seconds


MPRINT(OPTO_CTW):   proc corr cov data=_dtt1 outp=_covmat nosimple noprint;
MPRINT(OPTO_CTW):   var CPO FBNDX FCNTX FGRIX FLPSX FMAGX FOSFX FUSEX _GIC_;
MPRINT(OPTO_CTW):   run;

NOTE: The data set WORK._COVMAT has 21 observations and 11 variables.
NOTE: PROCEDURE CORR used:
      real time           0.01 seconds
      cpu time            0.01 seconds


MPRINT(DSV2MA):   proc contents data=pf.initds(drop = wgt cticker ticker) out=_m1(keep=name varnum ) short noprint;
MPRINT(DSV2MA):   run;

NOTE: The data set WORK._M1 has 1 observations and 2 variables.
NOTE: PROCEDURE CONTENTS used:
      real time           0.02 seconds
      cpu time            0.01 seconds


MPRINT(DS2MA):   proc sql noprint;
MPRINT(DS2MA):   select count(distinct name) into :_oth0 from _m1 where name ne '' ;
NOTE: PROCEDURE SQL used:
      real time           0.01 seconds
      cpu time            0.01 seconds


MPRINT(DS2MA):   proc sql noprint;
MPRINT(DS2MA):   select distinct name into :_oth1 - :_oth1 from _m1 where name ne '' order by name;
MPRINT(DSV2MA):  ;
NOTE: PROCEDURE SQL used:
      real time           0.00 seconds
      cpu time            0.01 seconds


MPRINT(TRASH):   proc datasets nolist lib=work;
MPRINT(TRASH):   delete _m1;
MPRINT(TRASH):   run;

NOTE: Deleting WORK._M1 (memtype=DATA).
MPRINT(DSV2MA):  ;
MPRINT(KKVNORM):  ;
MPRINT(KKVNORM):  ;
NOTE: PROCEDURE DATASETS used:
      real time           0.03 seconds
      cpu time            0.02 seconds


MPRINT(KKVNORM):   proc sql;
MPRINT(KKVNORM):   create table _kk3 as select cticker ,sum(wgt) as wgt from pf.initds group by cticker;
NOTE: Table WORK._KK3 created, with 1 rows and 2 columns.
```

```
NOTE: PROCEDURE SQL used:
      real time           0.01 seconds
      cpu time            0.00 seconds


MPRINT(KKVNORM):   proc sql;
MPRINT(KKVNORM):   create table _ctw as select a.cticker, a.ticker ,FNAME , a.wgt/s.wgt as wgt from _kk3 s, pf.initds a where
s.cticker = a.cticker order by a.cticker, a.ticker;
NOTE: Table WORK._CTW created, with 9 rows and 4 columns.

NOTE: PROCEDURE SQL used:
      real time           0.04 seconds
      cpu time            0.02 seconds


MPRINT(TRASH):   proc datasets nolist lib=work;
MPRINT(TRASH):   delete _kk3;
MPRINT(TRASH):   run;

NOTE: Deleting WORK._KK3 (memtype=DATA).
MPRINT(KKVNORM):  ;
MPRINT(CTWNORM):  ;
MPRINT(OPTO_CTW):  ;
Optimizing 1 / 1 : PF000021 lambda=5000 kappa=0.6 linear=PVALUEM minval=0

NOTE: PROCEDURE DATASETS used:
      real time           0.02 seconds
      cpu time            0.00 seconds


MPRINT(OPTO_CTW):   data _initval(keep=ticker wgt);
MPRINT(OPTO_CTW):   set _ctw(where=(cticker = "PF000021"));
MPRINT(OPTO_CTW):   run;

NOTE: There were 9 observations read from the data set WORK._CTW.
      WHERE cticker='PF000021';
NOTE: The data set WORK._INITVAL has 9 observations and 2 variables.
NOTE: DATA statement used:
      real time           0.01 seconds
      cpu time            0.00 seconds


MPRINT(OPTO_CTW):   proc sort data=_initval;
MPRINT(OPTO_CTW):   by ticker;
MPRINT(OPTO_CTW):   run;

NOTE: There were 9 observations read from the data set WORK._INITVAL.
NOTE: The data set WORK._INITVAL has 9 observations and 2 variables.
NOTE: PROCEDURE SORT used:
      real time           0.01 seconds
      cpu time            0.00 seconds


MPRINT(OPTO_CTW):   data _in_tv;
MPRINT(OPTO_CTW):   merge _initval(keep=ticker in=ina) product.asset_tv;
MPRINT(OPTO_CTW):   by ticker;
MPRINT(OPTO_CTW):   if ina;
MPRINT(OPTO_CTW):   run;
```

```
NOTE: There were 9 observations read from the data set WORK._INITVAL.
NOTE: There were 968 observations read from the data set PRODUCT.ASSET_TV.
NOTE: The data set WORK._IN_TV has 9 observations and 26 variables.
NOTE: DATA statement used:
      real time          0.03 seconds
      cpu time           0.02 seconds


MPRINT(DS2MA):   proc sql noprint;
MPRINT(DS2MA):   select count(distinct ticker) into :al0 from _in_tv where ticker ne '' ;
NOTE: PROCEDURE SQL used:
      real time          0.02 seconds
      cpu time           0.00 seconds


MPRINT(DS2MA):   proc sql noprint;
MPRINT(DS2MA):   select distinct ticker into :al1 - :al9 from _in_tv where ticker ne '' order by ticker;
MPRINT(QUAD):   ;
NOTE: PROCEDURE SQL used:
      real time          0.00 seconds
      cpu time           0.01 seconds


MPRINT(QUAD):   data pf.inquad(keep = _type_ _name_ CPO FBNDX FCNTX FGRIX FLPSX FMAGX FOSFX FUSEX _GIC_ _rhs_);
MPRINT(QUAD):   format _type_ $8. _name_ $8. _vname $8.;
MPRINT(QUAD):   set _covmat(where=(_type_ = 'COV' and _name_ in ("CPO" ,"FBNDX" ,"FCNTX" ,"FGRIX" ,"FLPSX" ,"FMAGX" ,"FOSFX"
,"FUSEX" ,"_GIC_")) ) end = eof;
MPRINT(QUAD):   retain _rhs_ . _vname '';
MPRINT(QUAD):   _type_ = 'QUAD';
MPRINT(QUAD):   CPO = CPO * 2 * 5000;
MPRINT(QUAD):   call vname(CPO,_vname);
MPRINT(QUAD):   if _name_ = _vname then CPO = CPO + 2 * 0.6;
MPRINT(QUAD):   FBNDX = FBNDX * 2 * 5000;
MPRINT(QUAD):   call vname(FBNDX,_vname);
MPRINT(QUAD):   if _name_ = _vname then FBNDX = FBNDX + 2 * 0.6;
MPRINT(QUAD):   FCNTX = FCNTX * 2 * 5000;
MPRINT(QUAD):   call vname(FCNTX,_vname);
MPRINT(QUAD):   if _name_ = _vname then FCNTX = FCNTX + 2 * 0.6;
MPRINT(QUAD):   FGRIX = FGRIX * 2 * 5000;
MPRINT(QUAD):   call vname(FGRIX,_vname);
MPRINT(QUAD):   if _name_ = _vname then FGRIX = FGRIX + 2 * 0.6;
MPRINT(QUAD):   FLPSX = FLPSX * 2 * 5000;
MPRINT(QUAD):   call vname(FLPSX,_vname);
MPRINT(QUAD):   if _name_ = _vname then FLPSX = FLPSX + 2 * 0.6;
MPRINT(QUAD):   FMAGX = FMAGX * 2 * 5000;
MPRINT(QUAD):   call vname(FMAGX,_vname);
MPRINT(QUAD):   if _name_ = _vname then FMAGX = FMAGX + 2 * 0.6;
MPRINT(QUAD):   FOSFX = FOSFX * 2 * 5000;
MPRINT(QUAD):   call vname(FOSFX,_vname);
MPRINT(QUAD):   if _name_ = _vname then FOSFX = FOSFX + 2 * 0.6;
MPRINT(QUAD):   FUSEX = FUSEX * 2 * 5000;
MPRINT(QUAD):   call vname(FUSEX,_vname);
MPRINT(QUAD):   if _name_ = _vname then FUSEX = FUSEX + 2 * 0.6;
MPRINT(QUAD):   _GIC_ = _GIC_ * 2 * 5000;
MPRINT(QUAD):   call vname(_GIC_,_vname);
MPRINT(QUAD):   if _name_ = _vname then _GIC_ = _GIC_ + 2 * 0.6;
MPRINT(QUAD):   output;
```

```
MPRINT(QUAD):   if eof then do;
MPRINT(QUAD):   CPO = 1;
MPRINT(QUAD):   FBNDX = 1;
MPRINT(QUAD):   FCNTX = 1;
MPRINT(QUAD):   FGRIX = 1;
MPRINT(QUAD):   FLPSX = 1;
MPRINT(QUAD):   FMAGX = 1;
MPRINT(QUAD):   FOSFX = 1;
MPRINT(QUAD):   FUSEX = 1;
MPRINT(QUAD):   _GIC_ = 1;
MPRINT(QUAD):   _type_ = 'eq';
MPRINT(QUAD):   _name_ = 'WGT';
MPRINT(QUAD):   _rhs_ = 1;
MPRINT(QUAD):   output;
MPRINT(QUAD):   end;
MPRINT(QUAD):   run;

NOTE: There were 9 observations read from the data set WORK._COVMAT.
      WHERE (_type_='COV') and _name_ in ('CPO', 'FBNDX', 'FCNTX', 'FGRIX', 'FLPSX', 'FMAGX', 'FOSFX', 'FUSEX', '_GIC_');
NOTE: The data set PF.INQUAD has 10 observations and 12 variables.
NOTE: DATA statement used:
      real time          0.05 seconds
      cpu time           0.04 seconds


MPRINT(QUAD):   proc sql noprint;
MPRINT(QUAD):   create table _q1 as select 'LINEAR  ' as _type_, . as _RHS_, ticker, PVALUEM as _linear from _in_tv ;
NOTE: Table WORK._Q1 created, with 9 rows and 4 columns.

NOTE: PROCEDURE SQL used:
      real time          0.01 seconds
      cpu time           0.00 seconds


MPRINT(QUAD):   proc transpose data=_q1 out=_q1;
MPRINT(QUAD):   var _linear;
MPRINT(QUAD):   id ticker;
MPRINT(QUAD):   by _type_ _RHS_ ;

NOTE: There were 9 observations read from the data set WORK._Q1.
NOTE: The data set WORK._Q1 has 1 observations and 12 variables.
NOTE: PROCEDURE TRANSPOSE used:
      real time          0.03 seconds
      cpu time           0.00 seconds


MPRINT(QUAD):   data pf.inquad;
MPRINT(QUAD):   set pf.inquad _q1;
MPRINT(QUAD):   run;

NOTE: There were 10 observations read from the data set PF.INQUAD.
NOTE: There were 1 observations read from the data set WORK._Q1.
NOTE: The data set PF.INQUAD has 11 observations and 12 variables.
NOTE: DATA statement used:
      real time          0.02 seconds
      cpu time           0.01 seconds
```

```
MPRINT(S2QCONST):   proc sort data=pf.sconst(keep = ticker maxwgt) out=_q3;
MPRINT(S2QCONST):   by ticker maxwgt;

NOTE: There were 4 observations read from the data set PF.SCONST.
NOTE: The data set WORK._Q3 has 4 observations and 2 variables.
NOTE: PROCEDURE SORT used:
      real time           0.02 seconds
      cpu time            0.00 seconds


MPRINT(S2QCONST):   data _q3;
MPRINT(S2QCONST):   set _q3;
MPRINT(S2QCONST):   by ticker maxwgt;
MPRINT(S2QCONST):   if first.ticker;

NOTE: There were 4 observations read from the data set WORK._Q3.
NOTE: The data set WORK._Q3 has 4 observations and 2 variables.
NOTE: DATA statement used:
      real time           0.01 seconds
      cpu time            0.00 seconds


MPRINT(S2QCONST):   proc transpose data=_q3 out=_q4;
MPRINT(S2QCONST):   var maxwgt;
MPRINT(S2QCONST):   id ticker;
NOTE: There were 4 observations read from the data set WORK._Q3.
NOTE: The data set WORK._Q4 has 1 observations and 5 variables.
NOTE: PROCEDURE TRANSPOSE used:
      real time           0.01 seconds
      cpu time            0.00 seconds


MPRINT(S2QCONST):   data _q4;
MPRINT(S2QCONST):   format _type_ $8. _name_ $8.;
MPRINT(S2QCONST):   set _q4;
MPRINT(S2QCONST):   _type_ = 'upperbd ';
MPRINT(S2QCONST):   _name_ = 'UPPER';
MPRINT(S2QCONST):   _RHS_ = .;
MPRINT(S2QCONST):   if CPO eq . then CPO = 1;
MPRINT(S2QCONST):   if FBNDX eq . then FBNDX = 1;
MPRINT(S2QCONST):   if FCNTX eq . then FCNTX = 1;
MPRINT(S2QCONST):   if FGRIX eq . then FGRIX = 1;
MPRINT(S2QCONST):   if FLPSX eq . then FLPSX = 1;
MPRINT(S2QCONST):   if FMAGX eq . then FMAGX = 1;
MPRINT(S2QCONST):   if FOSFX eq . then FOSFX = 1;
MPRINT(S2QCONST):   if FUSEX eq . then FUSEX = 1;
MPRINT(S2QCONST):   if _GIC_ eq . then _GIC_ = 1;
MPRINT(S2QCONST):   run;

NOTE: There were 1 observations read from the data set WORK._Q4.
NOTE: The data set WORK._Q4 has 1 observations and 14 variables.
NOTE: DATA statement used:
      real time           0.02 seconds
      cpu time            0.02 seconds


MPRINT(S2QCONST):   proc sort data=pf.sconst(keep = ticker minwgt) out=_q5;
MPRINT(S2QCONST):   by ticker descending minwgt;
```

```
NOTE: There were 4 observations read from the data set PF.SCONST.
NOTE: The data set WORK._Q5 has 4 observations and 2 variables.
NOTE: PROCEDURE SORT used:
      real time           0.02 seconds
      cpu time            0.00 seconds


MPRINT(S2QCONST):   data _q5;
MPRINT(S2QCONST):   set _q5;
MPRINT(S2QCONST):   by ticker descending minwgt;
MPRINT(S2QCONST):   if first.ticker;

NOTE: There were 4 observations read from the data set WORK._Q5.
NOTE: The data set WORK._Q5 has 4 observations and 2 variables.
NOTE: DATA statement used:
      real time           0.01 seconds
      cpu time            0.00 seconds


MPRINT(S2QCONST):   proc transpose data=_q5 out=_q6;
MPRINT(S2QCONST):   var minwgt;
MPRINT(S2QCONST):   id ticker;
NOTE: There were 4 observations read from the data set WORK._Q5.
NOTE: The data set WORK._Q6 has 1 observations and 5 variables.
NOTE: PROCEDURE TRANSPOSE used:
      real time           0.01 seconds
      cpu time            0.00 seconds


MPRINT(S2QCONST):   data _q6;
MPRINT(S2QCONST):   format _type_ $8. _name_ $8.;
MPRINT(S2QCONST):   set _q6;
MPRINT(S2QCONST):   _type_ = 'lowerbd ';
MPRINT(S2QCONST):   _name_ = 'LOWER';
MPRINT(S2QCONST):   _RHS_ = .;
MPRINT(S2QCONST):   if CPO eq . then CPO = 0;
MPRINT(S2QCONST):   if FBNDX eq . then FBNDX = 0;
MPRINT(S2QCONST):   if FCNTX eq . then FCNTX = 0;
MPRINT(S2QCONST):   if FGRIX eq . then FGRIX = 0;
MPRINT(S2QCONST):   if FLPSX eq . then FLPSX = 0;
MPRINT(S2QCONST):   if FMAGX eq . then FMAGX = 0;
MPRINT(S2QCONST):   if FOSFX eq . then FOSFX = 0;
MPRINT(S2QCONST):   if FUSEX eq . then FUSEX = 0;
MPRINT(S2QCONST):   if _GIC_ eq . then _GIC_ = 0;
MPRINT(S2QCONST):   run;

NOTE: There were 1 observations read from the data set WORK._Q6.
NOTE: The data set WORK._Q6 has 1 observations and 14 variables.
NOTE: DATA statement used:
      real time           0.02 seconds
      cpu time            0.02 seconds


MPRINT(S2QCONST):   data tsconst(keep = _type_ _name_ _rhs_ CPO FBNDX FCNTX FGRIX FLPSX FMAGX FOSFX FUSEX _GIC_);
MPRINT(S2QCONST):   set _q4 _q6;
MPRINT(S2QCONST):   run;
```

```
NOTE: There were 1 observations read from the data set WORK._Q4.
NOTE: There were 1 observations read from the data set WORK._Q6.
NOTE: The data set WORK.TSCONST has 2 observations and 12 variables.
NOTE: DATA statement used:
      real time          0.02 seconds
      cpu time           0.01 seconds


MPRINT(TRASH):   proc datasets nolist lib=work;
MPRINT(TRASH):   delete _q3 _q4 _q5 _q6;
MPRINT(TRASH):   run;

NOTE: Deleting WORK._Q3 (memtype=DATA).
NOTE: Deleting WORK._Q4 (memtype=DATA).
NOTE: Deleting WORK._Q5 (memtype=DATA).
NOTE: Deleting WORK._Q6 (memtype=DATA).
MPRINT(S2QCONST):  ;
MPRINT(QUAD):  ;
NOTE: PROCEDURE DATASETS used:
      real time          0.04 seconds
      cpu time           0.02 seconds


MPRINT(QUAD):   proc append base=pf.inquad data=tsconst force;
MPRINT(QUAD):   run;

NOTE: Appending WORK.TSCONST to PF.INQUAD.
NOTE: There were 2 observations read from the data set WORK.TSCONST.
NOTE: 2 observations added.
NOTE: The data set PF.INQUAD has 13 observations and 12 variables.
NOTE: PROCEDURE APPEND used:
      real time          0.01 seconds
      cpu time           0.01 seconds


MPRINT(QUAD):   proc transpose data=_initval out=_q1;
MPRINT(QUAD):   var wgt;
MPRINT(QUAD):   id ticker;
MPRINT(QUAD):   run;

NOTE: There were 9 observations read from the data set WORK._INITVAL.
NOTE: The data set WORK._Q1 has 1 observations and 10 variables.
NOTE: PROCEDURE TRANSPOSE used:
      real time          0.01 seconds
      cpu time           0.00 seconds


MPRINT(QUAD):   data _q1;
MPRINT(QUAD):   format _name_ $8.;
MPRINT(QUAD):   set _q1;
MPRINT(QUAD):   _type_ = 'PARMS   ';
MPRINT(QUAD):   _RHS_ = .;

NOTE: There were 1 observations read from the data set WORK._Q1.
NOTE: The data set WORK._Q1 has 1 observations and 12 variables.
NOTE: DATA statement used:
      real time          0.01 seconds
      cpu time           0.01 seconds
```

```
MPRINT(QUAD):   proc append base=pf.inquad data=_q1 force;
MPRINT(QUAD):   run;

NOTE: Appending WORK._Q1 to PF.INQUAD.
NOTE: There were 1 observations read from the data set WORK._Q1.
NOTE: 1 observations added.
NOTE: The data set PF.INQUAD has 14 observations and 12 variables.
NOTE: PROCEDURE APPEND used:
      real time          0.01 seconds
      cpu time           0.01 seconds


MPRINT(T2QCONST):   proc sql noprint;
MPRINT(T2QCONST):   select count(*) into :cnt from pf.adjconst;
NOTE: PROCEDURE SQL used:
      real time          0.00 seconds
      cpu time           0.00 seconds


MPRINT(T2QCONST):   proc sql noprint;
MPRINT(T2QCONST):   select var, minwgt,maxwgt into :vname1 - :vname6, :minwgt1 - :minwgt6, :maxwgt1 - :maxwgt6 from pf.adjconst;
NOTE: PROCEDURE SQL used:
      real time          0.00 seconds
      cpu time           0.01 seconds


MPRINT(T2QCONST):   data _q7;
MPRINT(T2QCONST):   format _name_ $8. ticker $8. wgt 15.8 _type_ $8. _rhs_ 15.8;
MPRINT(T2QCONST):   if 1=2;
MPRINT(T2QCONST):   _name_='';
MPRINT(T2QCONST):   ticker = '';
MPRINT(T2QCONST):   wgt = .;
MPRINT(T2QCONST):   _type_ = '';
MPRINT(T2QCONST):   _rhs_ = .;
MPRINT(T2QCONST):   run;

NOTE: The data set WORK._Q7 has 0 observations and 5 variables.
NOTE: DATA statement used:
      real time          0.01 seconds
      cpu time           0.01 seconds


MPRINT(T2QCONST):   data _q7;
MPRINT(T2QCONST):   set _q7 _in_tv(in=inb keep=ticker BONDS rename=(BONDS=wgt ));
MPRINT(T2QCONST):   if inb then do;
MPRINT(T2QCONST):   _name_ = "BONDS";
MPRINT(T2QCONST):   _type_ = 'ge';
MPRINT(T2QCONST):   _rhs_ = 0.01647;
MPRINT(T2QCONST):   end;
MPRINT(T2QCONST):   run;

NOTE: There were 0 observations read from the data set WORK._Q7.
NOTE: There were 9 observations read from the data set WORK._IN_TV.
NOTE: The data set WORK._Q7 has 9 observations and 5 variables.
NOTE: DATA statement used:
      real time          0.02 seconds
```

```
      cpu time            0.02 seconds


MPRINT(T2QCONST):   data _q7;
MPRINT(T2QCONST):   set _q7 _in_tv(in=inb keep=ticker BONDS rename=(BONDS=wgt));
MPRINT(T2QCONST):   if inb then do;
MPRINT(T2QCONST):   _name_ = "BONDS";
MPRINT(T2QCONST):   _type_ = 'le';
MPRINT(T2QCONST):   _rhs_ = 0.21647;
MPRINT(T2QCONST):   end;
MPRINT(T2QCONST):   run;

NOTE: There were 9 observations read from the data set WORK._Q7.
NOTE: There were 9 observations read from the data set WORK._IN_TV.
NOTE: The data set WORK._Q7 has 18 observations and 5 variables.
NOTE: DATA statement used:
      real time           0.02 seconds
      cpu time            0.02 seconds


MPRINT(T2QCONST):   data _q7;
MPRINT(T2QCONST):   set _q7 _in_tv(in=inb keep=ticker CASH rename=(CASH=wgt ));
MPRINT(T2QCONST):   if inb then do;
MPRINT(T2QCONST):   _name_ = "CASH";
MPRINT(T2QCONST):   _type_ = 'ge';
MPRINT(T2QCONST):   _rhs_ = -0.0735;
MPRINT(T2QCONST):   end;
MPRINT(T2QCONST):   run;

NOTE: There were 18 observations read from the data set WORK._Q7.
NOTE: There were 9 observations read from the data set WORK._IN_TV.
NOTE: The data set WORK._Q7 has 27 observations and 5 variables.
NOTE: DATA statement used:
      real time           0.02 seconds
      cpu time            0.01 seconds


MPRINT(T2QCONST):   data _q7;
MPRINT(T2QCONST):   set _q7 _in_tv(in=inb keep=ticker CASH rename=(CASH=wgt));
MPRINT(T2QCONST):   if inb then do;
MPRINT(T2QCONST):   _name_ = "CASH";
MPRINT(T2QCONST):   _type_ = 'le';
MPRINT(T2QCONST):   _rhs_ = 0.126496;
MPRINT(T2QCONST):   end;
MPRINT(T2QCONST):   run;

NOTE: There were 27 observations read from the data set WORK._Q7.
NOTE: There were 9 observations read from the data set WORK._IN_TV.
NOTE: The data set WORK._Q7 has 36 observations and 5 variables.
NOTE: DATA statement used:
      real time           0.02 seconds
      cpu time            0.01 seconds


MPRINT(T2QCONST):   data _q7;
MPRINT(T2QCONST):   set _q7 _in_tv(in=inb keep=ticker FSTOCKS rename=(FSTOCKS=wgt ));
MPRINT(T2QCONST):   if inb then do;
MPRINT(T2QCONST):   _name_ = "FSTOCKS";
```

```
MPRINT(T2QCONST):   _type_ = 'ge';
MPRINT(T2QCONST):   _rhs_ = -0.05657;
MPRINT(T2QCONST):   end;
MPRINT(T2QCONST):   run;

NOTE: There were 36 observations read from the data set WORK._Q7.
NOTE: There were 9 observations read from the data set WORK._IN_TV.
NOTE: The data set WORK._Q7 has 45 observations and 5 variables.
NOTE: DATA statement used:
      real time          0.02 seconds
      cpu time           0.01 seconds


MPRINT(T2QCONST):   data _q7;
MPRINT(T2QCONST):   set _q7 _in_tv(in=inb keep=ticker FSTOCKS rename=(FSTOCKS=wgt));
MPRINT(T2QCONST):   if inb then do;
MPRINT(T2QCONST):   _name_ = "FSTOCKS";
MPRINT(T2QCONST):   _type_ = 'le';
MPRINT(T2QCONST):   _rhs_ = 0.103428;
MPRINT(T2QCONST):   end;
MPRINT(T2QCONST):   run;

NOTE: There were 45 observations read from the data set WORK._Q7.
NOTE: There were 9 observations read from the data set WORK._IN_TV.
NOTE: The data set WORK._Q7 has 54 observations and 5 variables.
NOTE: DATA statement used:
      real time          0.02 seconds
      cpu time           0.01 seconds


MPRINT(T2QCONST):   data _q7;
MPRINT(T2QCONST):   set _q7 _in_tv(in=inb keep=ticker HYBONDS rename=(HYBONDS=wgt ));
MPRINT(T2QCONST):   if inb then do;
MPRINT(T2QCONST):   _name_ = "HYBONDS";
MPRINT(T2QCONST):   _type_ = 'ge';
MPRINT(T2QCONST):   _rhs_ = -0.1;
MPRINT(T2QCONST):   end;
MPRINT(T2QCONST):   run;

NOTE: There were 54 observations read from the data set WORK._Q7.
NOTE: There were 9 observations read from the data set WORK._IN_TV.
NOTE: The data set WORK._Q7 has 63 observations and 5 variables.
NOTE: DATA statement used:
      real time          0.02 seconds
      cpu time           0.01 seconds


MPRINT(T2QCONST):   data _q7;
MPRINT(T2QCONST):   set _q7 _in_tv(in=inb keep=ticker HYBONDS rename=(HYBONDS=wgt));
MPRINT(T2QCONST):   if inb then do;
MPRINT(T2QCONST):   _name_ = "HYBONDS";
MPRINT(T2QCONST):   _type_ = 'le';
MPRINT(T2QCONST):   _rhs_ = 0.1;
MPRINT(T2QCONST):   end;
MPRINT(T2QCONST):   run;

NOTE: There were 63 observations read from the data set WORK._Q7.
NOTE: There were 9 observations read from the data set WORK._IN_TV.
```

```
NOTE: The data set WORK._Q7 has 72 observations and 5 variables.
NOTE: DATA statement used:
      real time           0.01 seconds
      cpu time            0.02 seconds


MPRINT(T2QCONST):   data _q7;
MPRINT(T2QCONST):   set _q7 _in_tv(in=inb keep=ticker OTHER rename=(OTHER=wgt ));
MPRINT(T2QCONST):   if inb then do;
MPRINT(T2QCONST):   _name_ = "OTHER";
MPRINT(T2QCONST):   _type_ = 'ge';
MPRINT(T2QCONST):   _rhs_ = -0.1;
MPRINT(T2QCONST):   end;
MPRINT(T2QCONST):   run;

NOTE: There were 72 observations read from the data set WORK._Q7.
NOTE: There were 9 observations read from the data set WORK._IN_TV.
NOTE: The data set WORK._Q7 has 81 observations and 5 variables.
NOTE: DATA statement used:
      real time           0.02 seconds
      cpu time            0.02 seconds


MPRINT(T2QCONST):   data _q7;
MPRINT(T2QCONST):   set _q7 _in_tv(in=inb keep=ticker OTHER rename=(OTHER=wgt));
MPRINT(T2QCONST):   if inb then do;
MPRINT(T2QCONST):   _name_ = "OTHER";
MPRINT(T2QCONST):   _type_ = 'le';
MPRINT(T2QCONST):   _rhs_ = 0.1;
MPRINT(T2QCONST):   end;
MPRINT(T2QCONST):   run;

NOTE: There were 81 observations read from the data set WORK._Q7.
NOTE: There were 9 observations read from the data set WORK._IN_TV.
NOTE: The data set WORK._Q7 has 90 observations and 5 variables.
NOTE: DATA statement used:
      real time           0.02 seconds
      cpu time            0.01 seconds


MPRINT(T2QCONST):   data _q7;
MPRINT(T2QCONST):   set _q7 _in_tv(in=inb keep=ticker STOCKS rename=(STOCKS=wgt ));
MPRINT(T2QCONST):   if inb then do;
MPRINT(T2QCONST):   _name_ = "STOCKS";
MPRINT(T2QCONST):   _type_ = 'ge';
MPRINT(T2QCONST):   _rhs_ = 0.847033;
MPRINT(T2QCONST):   end;
MPRINT(T2QCONST):   run;

NOTE: There were 90 observations read from the data set WORK._Q7.
NOTE: There were 9 observations read from the data set WORK._IN_TV.
NOTE: The data set WORK._Q7 has 99 observations and 5 variables.
NOTE: DATA statement used:
      real time           0.01 seconds
      cpu time            0.01 seconds


MPRINT(T2QCONST):   data _q7;
```

```
MPRINT(T2QCONST):   set _q7 _in_tv(in=inb keep=ticker STOCKS rename=(STOCKS=wgt));
MPRINT(T2QCONST):   if inb then do;
MPRINT(T2QCONST):   _name_ = "STOCKS";
MPRINT(T2QCONST):   _type_ = 'le';
MPRINT(T2QCONST):   _rhs_ = 0.867033;
MPRINT(T2QCONST):   end;
MPRINT(T2QCONST):   run;

NOTE: There were 99 observations read from the data set WORK._Q7.
NOTE: There were 9 observations read from the data set WORK._IN_TV.
NOTE: The data set WORK._Q7 has 108 observations and 5 variables.
NOTE: DATA statement used:
      real time           0.01 seconds
      cpu time            0.01 seconds


MPRINT(T2QCONST):   proc sort data=_q7;
MPRINT(T2QCONST):   by _name_ _type_ _rhs_;
MPRINT(T2QCONST):   run;

NOTE: There were 108 observations read from the data set WORK._Q7.
NOTE: The data set WORK._Q7 has 108 observations and 5 variables.
NOTE: PROCEDURE SORT used:
      real time           0.01 seconds
      cpu time            0.01 seconds


MPRINT(T2QCONST):   proc transpose data=_q7 out=ttconst(drop=_label_);
MPRINT(T2QCONST):   by _name_ _type_ _rhs_;
MPRINT(T2QCONST):   id ticker;
MPRINT(T2QCONST):   var wgt;
MPRINT(T2QCONST):   run;

NOTE: There were 108 observations read from the data set WORK._Q7.
NOTE: The data set WORK.TTCONST has 12 observations and 12 variables.
NOTE: PROCEDURE TRANSPOSE used:
      real time           0.02 seconds
      cpu time            0.01 seconds


MPRINT(TRASH):   proc datasets nolist lib=work;
MPRINT(TRASH):   delete _q7;
MPRINT(TRASH):   run;

NOTE: Deleting WORK._Q7 (memtype=DATA).
MPRINT(T2QCONST):  ;
MPRINT(QUAD):  ;
NOTE: PROCEDURE DATASETS used:
      real time           0.01 seconds
      cpu time            0.01 seconds


MPRINT(QUAD):   proc append base=pf.inquad data=ttconst force;
MPRINT(QUAD):   run;

NOTE: Appending WORK.TTCONST to PF.INQUAD.
NOTE: There were 12 observations read from the data set WORK.TTCONST.
NOTE: 12 observations added.
```

```
NOTE: The data set PF.INQUAD has 26 observations and 12 variables.
NOTE: PROCEDURE APPEND used:
      real time          0.01 seconds
      cpu time           0.00 seconds


MPRINT(CALLNLP):   proc nlp inquad=pf.inquad outest=pf.outquad cov=2 ;
MPRINT(CALLNLP):   min;
MPRINT(CALLNLP):   parms CPO FBNDX FCNTX FGRIX FLPSX FMAGX FOSFX FUSEX _GIC_ ;
MPRINT(CALLNLP):   run;

NOTE: Your code contains 0 program statements.
NOTE: Analytic derivatives are used with a quadratic optimization problem.
WARNING: A total of 1 identical linear constraints are deleted.
NOTE: Initial value of parameter CPO is set randomly to 0.4825134098.
NOTE: Initial value of parameter FBNDX is set randomly to 0.781773595.
NOTE: Initial value of parameter FCNTX is set randomly to 0.5300982159.
NOTE: Initial value of parameter FGRIX is set randomly to 0.5869857937.
NOTE: Initial value of parameter FLPSX is set randomly to 0.3789469704.
NOTE: Initial value of parameter FMAGX is set randomly to 0.0491516348.
NOTE: Initial value of parameter FOSFX is set randomly to 0.5774327547.
NOTE: Initial value of parameter FUSEX is set randomly to 0.1838599677.
NOTE: Initial value of parameter _GIC_ is set randomly to 0.898012612.
NOTE: Initial point was changed to be feasible for boundary and linear constraints.
NOTE: ABSGCONV convergence criterion satisfied.
NOTE: There were 26 observations read from the data set PF.INQUAD.
NOTE: The data set PF.OUTQUAD has 63 observations and 14 variables.
NOTE: The PROCEDURE NLP printed pages 6-9.
NOTE: PROCEDURE NLP used:
      real time          0.13 seconds
      cpu time           0.06 seconds


MPRINT(CALLNLP):   proc sql noprint;
MPRINT(CALLNLP):   select _rhs_ into :objfnval from pf.outquad where _type_ = 'PARMS';
MPRINT(QUAD):  ;
NOTE: PROCEDURE SQL used:
      real time          0.00 seconds
      cpu time           0.01 seconds


MPRINT(TRASH):   proc datasets nolist lib=work;
MPRINT(TRASH):   delete _q1 _q2;
MPRINT(TRASH):   run;

NOTE: The file WORK._Q2 (memtype=DATA) was not found, but appears on a DELETE statement.
NOTE: Deleting WORK._Q1 (memtype=DATA).
MPRINT(QUAD):  ;
MPRINT(OPTO_CTW):  ;

NOTE: PROCEDURE DATASETS used:
      real time          0.02 seconds
      cpu time           0.02 seconds


MPRINT(Q2CTW):   proc sql;
MPRINT(Q2CTW):   select _rhs_ into :objfnval from pf.outquad where _type_ = 'PARMS';
NOTE: The PROCEDURE SQL printed page 10.
```

```
NOTE: PROCEDURE SQL used:
      real time           0.00 seconds
      cpu time            0.01 seconds


MPRINT(Q2CTW):   data _out_ctw(drop=_type_ _rhs_);
MPRINT(Q2CTW):   format cticker $8.;
MPRINT(Q2CTW):   set pf.outquad(drop= _name_ _iter_ _tech_ where=(_type_='PARMS'));
MPRINT(Q2CTW):   cticker = "PF000021";
MPRINT(Q2CTW):   call symput("OBJFNVAL",put(_rhs_,best.));
MPRINT(Q2CTW):   if _rhs_ ne . then do;
MPRINT(Q2CTW):   output;
MPRINT(Q2CTW):   call symput("OSTATUS","0");
MPRINT(Q2CTW):   end;
MPRINT(Q2CTW):   else do;
MPRINT(Q2CTW):   call symput("OSTATUS","1");
MPRINT(Q2CTW):   end;
MPRINT(Q2CTW):   run;

NOTE: There were 1 observations read from the data set PF.OUTQUAD.
      WHERE _type_='PARMS';
NOTE: The data set WORK._OUT_CTW has 1 observations and 10 variables.
NOTE: DATA statement used:
      real time           0.02 seconds
      cpu time            0.01 seconds


MPRINT(Q2CTW):   proc transpose data=_out_ctw out=_out_ctw(rename=(_name_=ticker col1=wgt));
MPRINT(Q2CTW):   by cticker;
MPRINT(Q2CTW):   run;

NOTE: There were 1 observations read from the data set WORK._OUT_CTW.
NOTE: The data set WORK._OUT_CTW has 9 observations and 3 variables.
NOTE: PROCEDURE TRANSPOSE used:
      real time           0.01 seconds
      cpu time            0.00 seconds


MPRINT(OPTO_CTW):  ;
MPRINT(OPTO_CTW):   data pf.ctw;
MPRINT(OPTO_CTW):   length cticker ticker $8;
MPRINT(OPTO_CTW):   set _out_ctw;
MPRINT(OPTO_CTW):   run;

NOTE: There were 9 observations read from the data set WORK._OUT_CTW.
NOTE: The data set PF.CTW has 9 observations and 3 variables.
NOTE: DATA statement used:
      real time           0.01 seconds
      cpu time            0.02 seconds


MPRINT(OPTO_PF):  ;
MPRINT(OPTO_PF):   proc sql;
MPRINT(OPTO_PF):   select min(wgt) into :mwgt from pf.ctw where wgt >0;
NOTE: The PROCEDURE SQL printed page 11.
NOTE: PROCEDURE SQL used:
      real time           0.00 seconds
      cpu time            0.00 seconds
```

```
MPRINT(OPTO_PF):   proc sql;
MPRINT(OPTO_PF):   create table initds as select a.*,b.atype,c.wgt as wgto from pf.initds a, product.asset_tv b, pf.ctw c where
a.ticker = b.ticker and a.ticker = c.ticker order by c.wgt desc, b.pvaluem;
NOTE: The query as specified involves ordering by an item that doesn't appear in its SELECT clause.
NOTE: Table WORK.INITDS created, with 9 rows and 6 columns.

NOTE: PROCEDURE SQL used:
      real time          0.04 seconds
      cpu time           0.03 seconds


MPRINT(OPTO_PF):   data initds(keep=cticker ticker wgt);
MPRINT(OPTO_PF):   set initds;
MPRINT(OPTO_PF):   retain ecount 0;
MPRINT(OPTO_PF):   retain fcount 0;
MPRINT(OPTO_PF):   if atype = 'EQUITY' then do;
MPRINT(OPTO_PF):   ecount = ecount + 1;
MPRINT(OPTO_PF):   if ecount <= 6 then output;
MPRINT(OPTO_PF):   end;
MPRINT(OPTO_PF):   else if atype = 'FOREIGN' then do;
MPRINT(OPTO_PF):   fcount = fcount + 1;
MPRINT(OPTO_PF):   if fcount <= 2 then output;
MPRINT(OPTO_PF):   end;
MPRINT(OPTO_PF):   else output;
MPRINT(OPTO_PF):   run;

NOTE: There were 9 observations read from the data set WORK.INITDS.
NOTE: The data set WORK.INITDS has 9 observations and 3 variables.
NOTE: DATA statement used:
      real time          0.02 seconds
      cpu time           0.02 seconds


MPRINT(DS2MA):   proc sql noprint;
MPRINT(DS2MA):   select count(distinct ticker) into :_mz0 from initds where ticker ne '' ;
NOTE: PROCEDURE SQL used:
      real time          0.02 seconds
      cpu time           0.01 seconds


MPRINT(DS2MA):   proc sql noprint;
MPRINT(DS2MA):   select distinct ticker into :_mz1 - :_mz9 from initds where ticker ne '' order by ticker;
MPRINT(DS2L):   ;
MPRINT(OPTO_PF):   ;
reduced universe = CPO FBNDX FCNTX FGRIX FLPSX FMAGX FOSFX FUSEX _GIC_
NOTE: PROCEDURE SQL used:
      real time          0.01 seconds
      cpu time           0.01 seconds


MPRINT(DS2MA):   proc sql noprint;
MPRINT(DS2MA):   select count(distinct ticker) into :_ma0 from initds where ticker ne '' ;
NOTE: PROCEDURE SQL used:
      real time          0.01 seconds
      cpu time           0.00 seconds
```

```
MPRINT(DS2MA):   proc sql noprint;
MPRINT(DS2MA):   select distinct ticker into :_ma1 - :_ma9 from initds where ticker ne '' order by ticker;
MPRINT(OPTO_CTW):  ;
NOTE: PROCEDURE SQL used:
      real time           0.00 seconds
      cpu time            0.01 seconds


MPRINT(DS2MA):   proc sql noprint;
MPRINT(DS2MA):   select count(distinct cticker) into :_p0 from initds where cticker ne '' ;
NOTE: PROCEDURE SQL used:
      real time           0.01 seconds
      cpu time            0.00 seconds


MPRINT(DS2MA):   proc sql noprint;
MPRINT(DS2MA):   select distinct cticker into :_p1 - :_p1 from initds where cticker ne '' order by cticker;
MPRINT(OPTO_CTW):  ;
NOTE: PROCEDURE SQL used:
      real time           0.00 seconds
      cpu time            0.00 seconds


MPRINT(OPTO_CTW):   proc sql noprint;
MPRINT(OPTO_CTW):   create table _ticks as select distinct ticker from initds;
NOTE: Table WORK._TICKS created, with 9 rows and 1 columns.

MPRINT(OPTO_CTW):   insert into _ticks(ticker) values ("_BENCH");
NOTE: 1 row was inserted into WORK._TICKS.

NOTE: PROCEDURE SQL used:
      real time           0.02 seconds
      cpu time            0.02 seconds


MPRINT(OPTO_CTW):   proc sort data=_ticks;
MPRINT(OPTO_CTW):   by ticker;
MPRINT(OPTO_CTW):   run;

NOTE: There were 10 observations read from the data set WORK._TICKS.
NOTE: The data set WORK._TICKS has 10 observations and 1 variables.
NOTE: PROCEDURE SORT used:
      real time           0.01 seconds
      cpu time            0.02 seconds


MPRINT(CLEANRET):   proc sql;
MPRINT(CLEANRET):   create table _rc2 as select distinct ticker from _ticks;
NOTE: Table WORK._RC2 created, with 10 rows and 1 columns.

NOTE: PROCEDURE SQL used:
      real time           0.01 seconds
      cpu time            0.01 seconds


MPRINT(CLEANRET):   data _rc2;
MPRINT(CLEANRET):   set _rc2 end=eof;
```

```
MPRINT(CLEANRET):   if _n_ eq 1 then call symput("_CRA",ticker);
MPRINT(CLEANRET):   if eof then call symput("_CRZ",ticker);
MPRINT(CLEANRET):   return = 0;
MPRINT(CLEANRET):   run;

NOTE: There were 10 observations read from the data set WORK._RC2.
NOTE: The data set WORK._RC2 has 10 observations and 2 variables.
NOTE: DATA statement used:
      real time          0.01 seconds
      cpu time           0.00 seconds


MPRINT(L2WTRAN):   proc transpose data=_rc2(keep=return ticker ) out=_rc2(keep= _numeric_);
MPRINT(L2WTRAN):   var return;
MPRINT(L2WTRAN):   id ticker;
MPRINT(L2WTRAN):   ;
MPRINT(L2WTRAN):   run;

NOTE: There were 10 observations read from the data set WORK._RC2.
NOTE: The data set WORK._RC2 has 1 observations and 10 variables.
NOTE: PROCEDURE TRANSPOSE used:
      real time          0.01 seconds
      cpu time           0.02 seconds


MPRINT(CLEANRET):   ;
MPRINT(CLEANRET):   data _dtt2(keep=date CPO -- _GIC_ );
MPRINT(CLEANRET):   merge _rc2 pf.mdl_dtt;
MPRINT(CLEANRET):   array __x CPO -- _GIC_ ;
MPRINT(CLEANRET):   do over __x;
MPRINT(CLEANRET):   if __x = . then __x = 0;
MPRINT(CLEANRET):   end;
MPRINT(CLEANRET):   output;
MPRINT(CLEANRET):   run;

NOTE: There were 1 observations read from the data set WORK._RC2.
NOTE: There were 36 observations read from the data set PF.MDL_DTT.
NOTE: The data set WORK._DTT2 has 36 observations and 11 variables.
NOTE: DATA statement used:
      real time          0.02 seconds
      cpu time           0.02 seconds


MPRINT(TRASH):   proc datasets nolist lib=work;
MPRINT(TRASH):   delete _rc2;
MPRINT(TRASH):   run;

NOTE: Deleting WORK._RC2 (memtype=DATA).
MPRINT(CLEANRET):   ;
MPRINT(OPTO_CTW):   ;

NOTE: PROCEDURE DATASETS used:
      real time          0.02 seconds
      cpu time           0.00 seconds


MPRINT(OPTO_CTW):   data _dtt1(drop=_name);
MPRINT(OPTO_CTW):   set _dtt2;
```

```
MPRINT(OPTO_CTW):   format _name $8.;
MPRINT(OPTO_CTW):   array __x _numeric_;
MPRINT(OPTO_CTW):   do over __x;
MPRINT(OPTO_CTW):   call vname(__x,_name);
MPRINT(OPTO_CTW):   if _name ne "_BENCH" and _name ne "DATE" then __x = __x - _BENCH;
MPRINT(OPTO_CTW):   end;

NOTE: There were 36 observations read from the data set WORK._DTT2.
NOTE: The data set WORK._DTT1 has 36 observations and 11 variables.
NOTE: DATA statement used:
      real time          0.01 seconds
      cpu time           0.02 seconds


MPRINT(OPTO_CTW):   proc corr cov data=_dtt1 outp=_covmat nosimple noprint;
MPRINT(OPTO_CTW):   var CPO FBNDX FCNTX FGRIX FLPSX FMAGX FOSFX FUSEX _GIC_;
MPRINT(OPTO_CTW):   run;

NOTE: The data set WORK._COVMAT has 21 observations and 11 variables.
NOTE: PROCEDURE CORR used:
      real time          0.01 seconds
      cpu time           0.02 seconds


MPRINT(DSV2MA):   proc contents data=initds(drop = wgt cticker ticker) out=_m1(keep=name varnum ) short noprint;
MPRINT(DSV2MA):   run;

NOTE: The data set WORK._M1 has 0 observations and 2 variables.
NOTE: PROCEDURE CONTENTS used:
      real time          0.02 seconds
      cpu time           0.01 seconds


MPRINT(DS2MA):   proc sql noprint;
MPRINT(DS2MA):   select count(distinct name) into :_oth0 from _m1 where name ne '' ;
MPRINT(DSV2MA):   ;
NOTE: PROCEDURE SQL used:
      real time          0.01 seconds
      cpu time           0.00 seconds


MPRINT(TRASH):   proc datasets nolist lib=work;
MPRINT(TRASH):   delete _m1;
MPRINT(TRASH):   run;

NOTE: Deleting WORK._M1 (memtype=DATA).
MPRINT(DSV2MA):   ;
MPRINT(KKVNORM):   ;
MPRINT(KKVNORM):   ;
NOTE: PROCEDURE DATASETS used:
      real time          0.02 seconds
      cpu time           0.02 seconds


MPRINT(KKVNORM):   proc sql;
MPRINT(KKVNORM):   create table _kk3 as select cticker ,sum(wgt) as wgt from initds group by cticker;
NOTE: Table WORK._KK3 created, with 1 rows and 2 columns.
```

```
NOTE: PROCEDURE SQL used:
      real time           0.01 seconds
      cpu time            0.01 seconds


MPRINT(KKVNORM):   proc sql;
MPRINT(KKVNORM):   create table _ctw as select a.cticker, a.ticker , a.wgt/s.wgt as wgt from _kk3 s, initds a where s.cticker =
a.cticker order by a.cticker, a.ticker;
NOTE: Table WORK._CTW created, with 9 rows and 3 columns.

NOTE: PROCEDURE SQL used:
      real time           0.03 seconds
      cpu time            0.02 seconds


MPRINT(TRASH):   proc datasets nolist lib=work;
MPRINT(TRASH):   delete _kk3;
MPRINT(TRASH):   run;

NOTE: Deleting WORK._KK3 (memtype=DATA).
MPRINT(KKVNORM):   ;
MPRINT(CTWNORM):   ;
MPRINT(OPTO_CTW):   ;
Optimizing 1 / 1 : PF000021 lambda=5000 kappa=0.6 linear=PVALUEM minval=0.05

NOTE: PROCEDURE DATASETS used:
      real time           0.02 seconds
      cpu time            0.01 seconds


MPRINT(OPTO_CTW):   data _initval(keep=ticker wgt);
MPRINT(OPTO_CTW):   set _ctw(where=(cticker = "PF000021"));
MPRINT(OPTO_CTW):   run;

NOTE: There were 9 observations read from the data set WORK._CTW.
      WHERE cticker='PF000021';
NOTE: The data set WORK._INITVAL has 9 observations and 2 variables.
NOTE: DATA statement used:
      real time           0.01 seconds
      cpu time            0.00 seconds


MPRINT(OPTO_CTW):   proc sort data=_initval;
MPRINT(OPTO_CTW):   by ticker;
MPRINT(OPTO_CTW):   run;

NOTE: There were 9 observations read from the data set WORK._INITVAL.
NOTE: The data set WORK._INITVAL has 9 observations and 2 variables.
NOTE: PROCEDURE SORT used:
      real time           0.01 seconds
      cpu time            0.01 seconds


MPRINT(OPTO_CTW):   data _in_tv;
MPRINT(OPTO_CTW):   merge _initval(keep=ticker in=ina) product.asset_tv;
MPRINT(OPTO_CTW):   by ticker;
MPRINT(OPTO_CTW):   if ina;
MPRINT(OPTO_CTW):   run;
```

```
NOTE: There were 9 observations read from the data set WORK._INITVAL.
NOTE: There were 968 observations read from the data set PRODUCT.ASSET_TV.
NOTE: The data set WORK._IN_TV has 9 observations and 26 variables.
NOTE: DATA statement used:
      real time           0.03 seconds
      cpu time            0.03 seconds


MPRINT(INITSTK):   proc sql noprint;
MPRINT(INITSTK):   create table stack (pos num, ticker char(8), lower num, upper num, prev num);
NOTE: Table WORK.STACK created, with 0 rows and 5 columns.
MPRINT(INITSTK):   create table stacko (pos num, ticker char(8), objval num, cur num, prev num, opt num, dist2min num);
NOTE: Table WORK.STACKO created, with 0 rows and 7 columns.
MPRINT(DISCOPT):  ;
NOTE: PROCEDURE SQL used:
      real time           0.03 seconds
      cpu time            0.01 seconds


MPRINT(DS2MA):   proc sql noprint;
MPRINT(DS2MA):   select count(distinct ticker) into :_qt0 from _in_tv where ticker ne '' ;
NOTE: PROCEDURE SQL used:
      real time           0.01 seconds
      cpu time            0.01 seconds


MPRINT(DS2MA):   proc sql noprint;
MPRINT(DS2MA):   select distinct ticker into :_qt1 - :_qt9 from _in_tv where ticker ne '' order by ticker;
MPRINT(DISCOPT):  ;
NOTE: PROCEDURE SQL used:
      real time           0.01 seconds
      cpu time            0.01 seconds


MPRINT(DS2MA):   proc sql noprint;
MPRINT(DS2MA):   select count(distinct ticker) into :al0 from _in_tv where ticker ne '' ;
NOTE: PROCEDURE SQL used:
      real time           0.01 seconds
      cpu time            0.01 seconds


MPRINT(DS2MA):   proc sql noprint;
MPRINT(DS2MA):   select distinct ticker into :al1 - :al9 from _in_tv where ticker ne '' order by ticker;
MPRINT(QUAD):  ;
NOTE: PROCEDURE SQL used:
      real time           0.00 seconds
      cpu time            0.01 seconds


MPRINT(QUAD):   data pf.inquad(keep = _type_ _name_ CPO FBNDX FCNTX FGRIX FLPSX FMAGX FOSFX FUSEX _GIC_ _rhs_);
MPRINT(QUAD):   format _type_ $8. _name_ $8. _vname $8.;
MPRINT(QUAD):   set _covmat(where=(_type_ = 'COV' and _name_ in ("CPO" ,"FBNDX" ,"FCNTX" ,"FGRIX" ,"FLPSX" ,"FMAGX" ,"FOSFX"
,"FUSEX" ,"_GIC_")) ) end = eof;
MPRINT(QUAD):   retain _rhs_ . _vname '';
MPRINT(QUAD):   _type_ = 'QUAD';
MPRINT(QUAD):   CPO = CPO * 2 * 5000;
MPRINT(QUAD):   call vname(CPO,_vname);
```

```
MPRINT(QUAD):   if _name_ = _vname then CPO = CPO + 2 * 0.6;
MPRINT(QUAD):   FBNDX = FBNDX * 2 * 5000;
MPRINT(QUAD):   call vname(FBNDX,_vname);
MPRINT(QUAD):   if _name_ = _vname then FBNDX = FBNDX + 2 * 0.6;
MPRINT(QUAD):   FCNTX = FCNTX * 2 * 5000;
MPRINT(QUAD):   call vname(FCNTX,_vname);
MPRINT(QUAD):   if _name_ = _vname then FCNTX = FCNTX + 2 * 0.6;
MPRINT(QUAD):   FGRIX = FGRIX * 2 * 5000;
MPRINT(QUAD):   call vname(FGRIX,_vname);
MPRINT(QUAD):   if _name_ = _vname then FGRIX = FGRIX + 2 * 0.6;
MPRINT(QUAD):   FLPSX = FLPSX * 2 * 5000;
MPRINT(QUAD):   call vname(FLPSX,_vname);
MPRINT(QUAD):   if _name_ = _vname then FLPSX = FLPSX + 2 * 0.6;
MPRINT(QUAD):   FMAGX = FMAGX * 2 * 5000;
MPRINT(QUAD):   call vname(FMAGX,_vname);
MPRINT(QUAD):   if _name_ = _vname then FMAGX = FMAGX + 2 * 0.6;
MPRINT(QUAD):   FOSFX = FOSFX * 2 * 5000;
MPRINT(QUAD):   call vname(FOSFX,_vname);
MPRINT(QUAD):   if _name_ = _vname then FOSFX = FOSFX + 2 * 0.6;
MPRINT(QUAD):   FUSEX = FUSEX * 2 * 5000;
MPRINT(QUAD):   call vname(FUSEX,_vname);
MPRINT(QUAD):   if _name_ = _vname then FUSEX = FUSEX + 2 * 0.6;
MPRINT(QUAD):   _GIC_ = _GIC_ * 2 * 5000;
MPRINT(QUAD):   call vname(_GIC_,_vname);
MPRINT(QUAD):   if _name_ = _vname then _GIC_ = _GIC_ + 2 * 0.6;
MPRINT(QUAD):   output;
MPRINT(QUAD):   if eof then do;
MPRINT(QUAD):   CPO = 1;
MPRINT(QUAD):   FBNDX = 1;
MPRINT(QUAD):   FCNTX = 1;
MPRINT(QUAD):   FGRIX = 1;
MPRINT(QUAD):   FLPSX = 1;
MPRINT(QUAD):   FMAGX = 1;
MPRINT(QUAD):   FOSFX = 1;
MPRINT(QUAD):   FUSEX = 1;
MPRINT(QUAD):   _GIC_ = 1;
MPRINT(QUAD):   _type_ = 'eq';
MPRINT(QUAD):   _name_ = 'WGT';
MPRINT(QUAD):   _rhs_ = 1;
MPRINT(QUAD):   output;
MPRINT(QUAD):   end;
MPRINT(QUAD):   run;

NOTE: There were 9 observations read from the data set WORK._COVMAT.
      WHERE (_type_='COV') and _name_ in ('CPO', 'FBNDX', 'FCNTX', 'FGRIX', 'FLPSX', 'FMAGX', 'FOSFX', 'FUSEX', '_GIC_');
NOTE: The data set PF.INQUAD has 10 observations and 12 variables.
NOTE: DATA statement used:
      real time          0.06 seconds
      cpu time           0.06 seconds


MPRINT(QUAD):   proc sql noprint;
MPRINT(QUAD):   create table _q1 as select 'LINEAR  ' as _type_, . as _RHS_, ticker, PVALUEM as _linear from _in_tv ;
NOTE: Table WORK._Q1 created, with 9 rows and 4 columns.

NOTE: PROCEDURE SQL used:
      real time          0.01 seconds
      cpu time           0.00 seconds
```

```
MPRINT(QUAD):   proc transpose data=_q1 out=_q1;
MPRINT(QUAD):   var _linear;
MPRINT(QUAD):   id ticker;
MPRINT(QUAD):   by _type_ _RHS_ ;

NOTE: There were 9 observations read from the data set WORK._Q1.
NOTE: The data set WORK._Q1 has 1 observations and 12 variables.
NOTE: PROCEDURE TRANSPOSE used:
      real time          0.01 seconds
      cpu time           0.01 seconds


MPRINT(QUAD):   data pf.inquad;
MPRINT(QUAD):   set pf.inquad _q1;
MPRINT(QUAD):   run;

NOTE: There were 10 observations read from the data set PF.INQUAD.
NOTE: There were 1 observations read from the data set WORK._Q1.
NOTE: The data set PF.INQUAD has 11 observations and 12 variables.
NOTE: DATA statement used:
      real time          0.02 seconds
      cpu time           0.00 seconds


MPRINT(S2QCONST):   proc sort data=pf.sconst(keep = ticker maxwgt) out=_q3;
MPRINT(S2QCONST):   by ticker maxwgt;

NOTE: There were 4 observations read from the data set PF.SCONST.
NOTE: The data set WORK._Q3 has 4 observations and 2 variables.
NOTE: PROCEDURE SORT used:
      real time          0.02 seconds
      cpu time           0.01 seconds


MPRINT(S2QCONST):   data _q3;
MPRINT(S2QCONST):   set _q3;
MPRINT(S2QCONST):   by ticker maxwgt;
MPRINT(S2QCONST):   if first.ticker;

NOTE: There were 4 observations read from the data set WORK._Q3.
NOTE: The data set WORK._Q3 has 4 observations and 2 variables.
NOTE: DATA statement used:
      real time          0.02 seconds
      cpu time           0.01 seconds


MPRINT(S2QCONST):   proc transpose data=_q3 out=_q4;
MPRINT(S2QCONST):   var maxwgt;
MPRINT(S2QCONST):   id ticker;
NOTE: There were 4 observations read from the data set WORK._Q3.
NOTE: The data set WORK._Q4 has 1 observations and 5 variables.
NOTE: PROCEDURE TRANSPOSE used:
      real time          0.01 seconds
      cpu time           0.00 seconds
```

```
MPRINT(S2QCONST):   data _q4;
MPRINT(S2QCONST):   format _type_ $8. _name_ $8.;
MPRINT(S2QCONST):   set _q4;
MPRINT(S2QCONST):   _type_ = 'upperbd ';
MPRINT(S2QCONST):   _name_ = 'UPPER';
MPRINT(S2QCONST):   _RHS_ = .;
MPRINT(S2QCONST):   if CPO eq . then CPO = 1;
MPRINT(S2QCONST):   if FBNDX eq . then FBNDX = 1;
MPRINT(S2QCONST):   if FCNTX eq . then FCNTX = 1;
MPRINT(S2QCONST):   if FGRIX eq . then FGRIX = 1;
MPRINT(S2QCONST):   if FLPSX eq . then FLPSX = 1;
MPRINT(S2QCONST):   if FMAGX eq . then FMAGX = 1;
MPRINT(S2QCONST):   if FOSFX eq . then FOSFX = 1;
MPRINT(S2QCONST):   if FUSEX eq . then FUSEX = 1;
MPRINT(S2QCONST):   if _GIC_ eq . then _GIC_ = 1;
MPRINT(S2QCONST):   run;

NOTE: There were 1 observations read from the data set WORK._Q4.
NOTE: The data set WORK._Q4 has 1 observations and 14 variables.
NOTE: DATA statement used:
      real time           0.02 seconds
      cpu time            0.01 seconds


MPRINT(S2QCONST):   proc sort data=pf.sconst(keep = ticker minwgt) out=_q5;
MPRINT(S2QCONST):   by ticker descending minwgt;

NOTE: There were 4 observations read from the data set PF.SCONST.
NOTE: The data set WORK._Q5 has 4 observations and 2 variables.
NOTE: PROCEDURE SORT used:
      real time           0.03 seconds
      cpu time            0.00 seconds


MPRINT(S2QCONST):   data _q5;
MPRINT(S2QCONST):   set _q5;
MPRINT(S2QCONST):   by ticker descending minwgt;
MPRINT(S2QCONST):   if first.ticker;

NOTE: There were 4 observations read from the data set WORK._Q5.
NOTE: The data set WORK._Q5 has 4 observations and 2 variables.
NOTE: DATA statement used:
      real time           0.01 seconds
      cpu time            0.02 seconds


MPRINT(S2QCONST):   proc transpose data=_q5 out=_q6;
MPRINT(S2QCONST):   var minwgt;
MPRINT(S2QCONST):   id ticker;
NOTE: There were 4 observations read from the data set WORK._Q5.
NOTE: The data set WORK._Q6 has 1 observations and 5 variables.
NOTE: PROCEDURE TRANSPOSE used:
      real time           0.01 seconds
      cpu time            0.00 seconds


MPRINT(S2QCONST):   data _q6;
MPRINT(S2QCONST):   format _type_ $8. _name_ $8.;
```

```
MPRINT(S2QCONST):   set _q6;
MPRINT(S2QCONST):   _type_ = 'lowerbd ';
MPRINT(S2QCONST):   _name_ = 'LOWER';
MPRINT(S2QCONST):   _RHS_ = .;
MPRINT(S2QCONST):   if CPO eq . then CPO = 0;
MPRINT(S2QCONST):   if FBNDX eq . then FBNDX = 0;
MPRINT(S2QCONST):   if FCNTX eq . then FCNTX = 0;
MPRINT(S2QCONST):   if FGRIX eq . then FGRIX = 0;
MPRINT(S2QCONST):   if FLPSX eq . then FLPSX = 0;
MPRINT(S2QCONST):   if FMAGX eq . then FMAGX = 0;
MPRINT(S2QCONST):   if FOSFX eq . then FOSFX = 0;
MPRINT(S2QCONST):   if FUSEX eq . then FUSEX = 0;
MPRINT(S2QCONST):   if _GIC_ eq . then _GIC_ = 0;
MPRINT(S2QCONST):   run;

NOTE: There were 1 observations read from the data set WORK._Q6.
NOTE: The data set WORK._Q6 has 1 observations and 14 variables.
NOTE: DATA statement used:
      real time           0.02 seconds
      cpu time            0.02 seconds


MPRINT(S2QCONST):   data tsconst(keep = _type_ _name_ _rhs_ CPO FBNDX FCNTX FGRIX FLPSX FMAGX FOSFX FUSEX _GIC_);
MPRINT(S2QCONST):   set _q4 _q6;
MPRINT(S2QCONST):   run;

NOTE: There were 1 observations read from the data set WORK._Q4.
NOTE: There were 1 observations read from the data set WORK._Q6.
NOTE: The data set WORK.TSCONST has 2 observations and 12 variables.
NOTE: DATA statement used:
      real time           0.02 seconds
      cpu time            0.02 seconds


MPRINT(TRASH):   proc datasets nolist lib=work;
MPRINT(TRASH):   delete _q3 _q4 _q5 _q6;
MPRINT(TRASH):   run;

NOTE: Deleting WORK._Q3 (memtype=DATA).
NOTE: Deleting WORK._Q4 (memtype=DATA).
NOTE: Deleting WORK._Q5 (memtype=DATA).
NOTE: Deleting WORK._Q6 (memtype=DATA).
MPRINT(S2QCONST):   ;
MPRINT(QUAD):   ;
NOTE: PROCEDURE DATASETS used:
      real time           0.04 seconds
      cpu time            0.00 seconds


MPRINT(QUAD):   proc append base=pf.inquad data=tsconst force;
MPRINT(QUAD):   run;

NOTE: Appending WORK.TSCONST to PF.INQUAD.
NOTE: There were 2 observations read from the data set WORK.TSCONST.
NOTE: 2 observations added.
NOTE: The data set PF.INQUAD has 13 observations and 12 variables.
NOTE: PROCEDURE APPEND used:
      real time           0.01 seconds
```

```
      cpu time            0.00 seconds

MPRINT(QUAD):   proc transpose data=_initval out=_q1;
MPRINT(QUAD):   var wgt;
MPRINT(QUAD):   id ticker;
MPRINT(QUAD):   run;

NOTE: There were 9 observations read from the data set WORK._INITVAL.
NOTE: The data set WORK._Q1 has 1 observations and 10 variables.
NOTE: PROCEDURE TRANSPOSE used:
      real time           0.01 seconds
      cpu time            0.00 seconds


MPRINT(QUAD):   data _q1;
MPRINT(QUAD):   format _name_ $8.;
MPRINT(QUAD):   set _q1;
MPRINT(QUAD):   _type_ = 'PARMS   ';
MPRINT(QUAD):   _RHS_ = .;

NOTE: There were 1 observations read from the data set WORK._Q1.
NOTE: The data set WORK._Q1 has 1 observations and 12 variables.
NOTE: DATA statement used:
      real time           0.01 seconds
      cpu time            0.02 seconds


MPRINT(QUAD):   proc append base=pf.inquad data=_q1 force;
MPRINT(QUAD):   run;

NOTE: Appending WORK._Q1 to PF.INQUAD.
NOTE: There were 1 observations read from the data set WORK._Q1.
NOTE: 1 observations added.
NOTE: The data set PF.INQUAD has 14 observations and 12 variables.
NOTE: PROCEDURE APPEND used:
      real time           0.01 seconds
      cpu time            0.00 seconds


MPRINT(T2QCONST):   proc sql noprint;
MPRINT(T2QCONST):   select count(*) into :cnt from pf.adjconst;
NOTE: PROCEDURE SQL used:
      real time           0.00 seconds
      cpu time            0.00 seconds


MPRINT(T2QCONST):   proc sql noprint;
MPRINT(T2QCONST):   select var, minwgt,maxwgt into :vname1 - :vname6, :minwgt1 - :minwgt6, :maxwgt1 - :maxwgt6 from pf.adjconst;
NOTE: PROCEDURE SQL used:
      real time           0.00 seconds
      cpu time            0.01 seconds


MPRINT(T2QCONST):   data _q7;
MPRINT(T2QCONST):   format _name_ $8. ticker $8. wgt 15.8 _type_ $8. _rhs_ 15.8;
MPRINT(T2QCONST):   if 1=2;
MPRINT(T2QCONST):   _name_='';
```

```
MPRINT(T2QCONST):   ticker = '';
MPRINT(T2QCONST):   wgt = .;
MPRINT(T2QCONST):   _type_ = '';
MPRINT(T2QCONST):   _rhs_ = .;
MPRINT(T2QCONST):   run;

NOTE: The data set WORK._Q7 has 0 observations and 5 variables.
NOTE: DATA statement used:
      real time          0.01 seconds
      cpu time           0.01 seconds


MPRINT(T2QCONST):   data _q7;
MPRINT(T2QCONST):   set _q7 _in_tv(in=inb keep=ticker BONDS rename=(BONDS=wgt ));
MPRINT(T2QCONST):   if inb then do;
MPRINT(T2QCONST):   _name_ = "BONDS";
MPRINT(T2QCONST):   _type_ = 'ge';
MPRINT(T2QCONST):   _rhs_ = 0.01647;
MPRINT(T2QCONST):   end;
MPRINT(T2QCONST):   run;

NOTE: There were 0 observations read from the data set WORK._Q7.
NOTE: There were 9 observations read from the data set WORK._IN_TV.
NOTE: The data set WORK._Q7 has 9 observations and 5 variables.
NOTE: DATA statement used:
      real time          0.02 seconds
      cpu time           0.01 seconds


MPRINT(T2QCONST):   data _q7;
MPRINT(T2QCONST):   set _q7 _in_tv(in=inb keep=ticker BONDS rename=(BONDS=wgt));
MPRINT(T2QCONST):   if inb then do;
MPRINT(T2QCONST):   _name_ = "BONDS";
MPRINT(T2QCONST):   _type_ = 'le';
MPRINT(T2QCONST):   _rhs_ = 0.21647;
MPRINT(T2QCONST):   end;
MPRINT(T2QCONST):   run;

NOTE: There were 9 observations read from the data set WORK._Q7.
NOTE: There were 9 observations read from the data set WORK._IN_TV.
NOTE: The data set WORK._Q7 has 18 observations and 5 variables.
NOTE: DATA statement used:
      real time          0.02 seconds
      cpu time           0.02 seconds


MPRINT(T2QCONST):   data _q7;
MPRINT(T2QCONST):   set _q7 _in_tv(in=inb keep=ticker CASH rename=(CASH=wgt ));
MPRINT(T2QCONST):   if inb then do;
MPRINT(T2QCONST):   _name_ = "CASH";
MPRINT(T2QCONST):   _type_ = 'ge';
MPRINT(T2QCONST):   _rhs_ = -0.0735;
MPRINT(T2QCONST):   end;
MPRINT(T2QCONST):   run;

NOTE: There were 18 observations read from the data set WORK._Q7.
NOTE: There were 9 observations read from the data set WORK._IN_TV.
NOTE: The data set WORK._Q7 has 27 observations and 5 variables.
```

```
NOTE: DATA statement used:
      real time           0.02 seconds
      cpu time            0.01 seconds


MPRINT(T2QCONST):   data _q7;
MPRINT(T2QCONST):   set _q7 _in_tv(in=inb keep=ticker CASH rename=(CASH=wgt));
MPRINT(T2QCONST):   if inb then do;
MPRINT(T2QCONST):   _name_ = "CASH";
MPRINT(T2QCONST):   _type_ = 'le';
MPRINT(T2QCONST):   _rhs_ = 0.126496;
MPRINT(T2QCONST):   end;
MPRINT(T2QCONST):   run;

NOTE: There were 27 observations read from the data set WORK._Q7.
NOTE: There were 9 observations read from the data set WORK._IN_TV.
NOTE: The data set WORK._Q7 has 36 observations and 5 variables.
NOTE: DATA statement used:
      real time           0.02 seconds
      cpu time            0.02 seconds


MPRINT(T2QCONST):   data _q7;
MPRINT(T2QCONST):   set _q7 _in_tv(in=inb keep=ticker FSTOCKS rename=(FSTOCKS=wgt ));
MPRINT(T2QCONST):   if inb then do;
MPRINT(T2QCONST):   _name_ = "FSTOCKS";
MPRINT(T2QCONST):   _type_ = 'ge';
MPRINT(T2QCONST):   _rhs_ = -0.05657;
MPRINT(T2QCONST):   end;
MPRINT(T2QCONST):   run;

NOTE: There were 36 observations read from the data set WORK._Q7.
NOTE: There were 9 observations read from the data set WORK._IN_TV.
NOTE: The data set WORK._Q7 has 45 observations and 5 variables.
NOTE: DATA statement used:
      real time           0.02 seconds
      cpu time            0.01 seconds


MPRINT(T2QCONST):   data _q7;
MPRINT(T2QCONST):   set _q7 _in_tv(in=inb keep=ticker FSTOCKS rename=(FSTOCKS=wgt));
MPRINT(T2QCONST):   if inb then do;
MPRINT(T2QCONST):   _name_ = "FSTOCKS";
MPRINT(T2QCONST):   _type_ = 'le';
MPRINT(T2QCONST):   _rhs_ = 0.103428;
MPRINT(T2QCONST):   end;
MPRINT(T2QCONST):   run;

NOTE: There were 45 observations read from the data set WORK._Q7.
NOTE: There were 9 observations read from the data set WORK._IN_TV.
NOTE: The data set WORK._Q7 has 54 observations and 5 variables.
NOTE: DATA statement used:
      real time           0.02 seconds
      cpu time            0.01 seconds


MPRINT(T2QCONST):   data _q7;
MPRINT(T2QCONST):   set _q7 _in_tv(in=inb keep=ticker HYBONDS rename=(HYBONDS=wgt ));
```

```
MPRINT(T2QCONST):   if inb then do;
MPRINT(T2QCONST):   _name_ = "HYBONDS";
MPRINT(T2QCONST):   _type_ = 'ge';
MPRINT(T2QCONST):   _rhs_ = -0.1;
MPRINT(T2QCONST):   end;
MPRINT(T2QCONST):   run;

NOTE: There were 54 observations read from the data set WORK._Q7.
NOTE: There were 9 observations read from the data set WORK._IN_TV.
NOTE: The data set WORK._Q7 has 63 observations and 5 variables.
NOTE: DATA statement used:
      real time          0.02 seconds
      cpu time           0.02 seconds


MPRINT(T2QCONST):   data _q7;
MPRINT(T2QCONST):   set _q7 _in_tv(in=inb keep=ticker HYBONDS rename=(HYBONDS=wgt));
MPRINT(T2QCONST):   if inb then do;
MPRINT(T2QCONST):   _name_ = "HYBONDS";
MPRINT(T2QCONST):   _type_ = 'le';
MPRINT(T2QCONST):   _rhs_ = 0.1;
MPRINT(T2QCONST):   end;
MPRINT(T2QCONST):   run;

NOTE: There were 63 observations read from the data set WORK._Q7.
NOTE: There were 9 observations read from the data set WORK._IN_TV.
NOTE: The data set WORK._Q7 has 72 observations and 5 variables.
NOTE: DATA statement used:
      real time          0.01 seconds
      cpu time           0.02 seconds


MPRINT(T2QCONST):   data _q7;
MPRINT(T2QCONST):   set _q7 _in_tv(in=inb keep=ticker OTHER rename=(OTHER=wgt ));
MPRINT(T2QCONST):   if inb then do;
MPRINT(T2QCONST):   _name_ = "OTHER";
MPRINT(T2QCONST):   _type_ = 'ge';
MPRINT(T2QCONST):   _rhs_ = -0.1;
MPRINT(T2QCONST):   end;
MPRINT(T2QCONST):   run;

NOTE: There were 72 observations read from the data set WORK._Q7.
NOTE: There were 9 observations read from the data set WORK._IN_TV.
NOTE: The data set WORK._Q7 has 81 observations and 5 variables.
NOTE: DATA statement used:
      real time          0.02 seconds
      cpu time           0.01 seconds


MPRINT(T2QCONST):   data _q7;
MPRINT(T2QCONST):   set _q7 _in_tv(in=inb keep=ticker OTHER rename=(OTHER=wgt));
MPRINT(T2QCONST):   if inb then do;
MPRINT(T2QCONST):   _name_ = "OTHER";
MPRINT(T2QCONST):   _type_ = 'le';
MPRINT(T2QCONST):   _rhs_ = 0.1;
MPRINT(T2QCONST):   end;
MPRINT(T2QCONST):   run;
```

```
NOTE: There were 81 observations read from the data set WORK._Q7.
NOTE: There were 9 observations read from the data set WORK._IN_TV.
NOTE: The data set WORK._Q7 has 90 observations and 5 variables.
NOTE: DATA statement used:
      real time           0.02 seconds
      cpu time            0.01 seconds


MPRINT(T2QCONST):   data _q7;
MPRINT(T2QCONST):   set _q7 _in_tv(in=inb keep=ticker STOCKS rename=(STOCKS=wgt ));
MPRINT(T2QCONST):   if inb then do;
MPRINT(T2QCONST):   _name_ = "STOCKS";
MPRINT(T2QCONST):   _type_ = 'ge';
MPRINT(T2QCONST):   _rhs_ = 0.847033;
MPRINT(T2QCONST):   end;
MPRINT(T2QCONST):   run;

NOTE: There were 90 observations read from the data set WORK._Q7.
NOTE: There were 9 observations read from the data set WORK._IN_TV.
NOTE: The data set WORK._Q7 has 99 observations and 5 variables.
NOTE: DATA statement used:
      real time           0.02 seconds
      cpu time            0.02 seconds


MPRINT(T2QCONST):   data _q7;
MPRINT(T2QCONST):   set _q7 _in_tv(in=inb keep=ticker STOCKS rename=(STOCKS=wgt));
MPRINT(T2QCONST):   if inb then do;
MPRINT(T2QCONST):   _name_ = "STOCKS";
MPRINT(T2QCONST):   _type_ = 'le';
MPRINT(T2QCONST):   _rhs_ = 0.867033;
MPRINT(T2QCONST):   end;
MPRINT(T2QCONST):   run;

NOTE: There were 99 observations read from the data set WORK._Q7.
NOTE: There were 9 observations read from the data set WORK._IN_TV.
NOTE: The data set WORK._Q7 has 108 observations and 5 variables.
NOTE: DATA statement used:
      real time           0.01 seconds
      cpu time            0.02 seconds


MPRINT(T2QCONST):   proc sort data=_q7;
MPRINT(T2QCONST):   by _name_ _type_ _rhs_;
MPRINT(T2QCONST):   run;

NOTE: There were 108 observations read from the data set WORK._Q7.
NOTE: The data set WORK._Q7 has 108 observations and 5 variables.
NOTE: PROCEDURE SORT used:
      real time           0.01 seconds
      cpu time            0.00 seconds


MPRINT(T2QCONST):   proc transpose data=_q7 out=ttconst(drop=_label_);
MPRINT(T2QCONST):   by _name_ _type_ _rhs_;
MPRINT(T2QCONST):   id ticker;
MPRINT(T2QCONST):   var wgt;
MPRINT(T2QCONST):   run;
```

```
NOTE: There were 108 observations read from the data set WORK._Q7.
NOTE: The data set WORK.TTCONST has 12 observations and 12 variables.
NOTE: PROCEDURE TRANSPOSE used:
      real time             0.02 seconds
      cpu time              0.01 seconds


MPRINT(TRASH):   proc datasets nolist lib=work;
MPRINT(TRASH):   delete _q7;
MPRINT(TRASH):   run;

NOTE: Deleting WORK._Q7 (memtype=DATA).
MPRINT(T2QCONST):   ;
MPRINT(QUAD):   ;
NOTE: PROCEDURE DATASETS used:
      real time             0.02 seconds
      cpu time              0.02 seconds


MPRINT(QUAD):   proc append base=pf.inquad data=ttconst force;
MPRINT(QUAD):   run;

NOTE: Appending WORK.TTCONST to PF.INQUAD.
NOTE: There were 12 observations read from the data set WORK.TTCONST.
NOTE: 12 observations added.
NOTE: The data set PF.INQUAD has 26 observations and 12 variables.
NOTE: PROCEDURE APPEND used:
      real time             0.01 seconds
      cpu time              0.01 seconds


MPRINT(TRASH):   proc datasets nolist lib=work;
MPRINT(TRASH):   delete _q1 _q2;
MPRINT(TRASH):   run;

NOTE: The file WORK._Q2 (memtype=DATA) was not found, but appears on a DELETE statement.
NOTE: Deleting WORK._Q1 (memtype=DATA).
MPRINT(QUAD):   ;
MPRINT(DISCOPT):   ;
NOTE: PROCEDURE DATASETS used:
      real time             0.02 seconds
      cpu time              0.01 seconds


MPRINT(DISCOPT):   proc transpose data=pf.inquad(drop= where=(_name_="LOWER")) out=l(rename=(_name_=ticker));
NOTE: There were 1 observations read from the data set PF.INQUAD.
      WHERE _name_='LOWER';
NOTE: The data set WORK.L has 10 observations and 2 variables.
NOTE: PROCEDURE TRANSPOSE used:
      real time             0.02 seconds
      cpu time              0.00 seconds


MPRINT(DISCOPT):   proc transpose data=pf.inquad(drop= where=(_name_="UPPER")) out=u(rename=(_name_=ticker));
NOTE: There were 1 observations read from the data set PF.INQUAD.
      WHERE _name_='UPPER';
NOTE: The data set WORK.U has 10 observations and 2 variables.
```

```
NOTE: PROCEDURE TRANSPOSE used:
      real time             0.02 seconds
      cpu time              0.01 seconds


MPRINT(DISCOPT):   proc sort data=u;
MPRINT(DISCOPT):   by ticker;
NOTE: There were 10 observations read from the data set WORK.U.
NOTE: The data set WORK.U has 10 observations and 2 variables.
NOTE: PROCEDURE SORT used:
      real time             0.01 seconds
      cpu time              0.00 seconds


MPRINT(DISCOPT):   proc sort data=l;
MPRINT(DISCOPT):   by ticker;
MPRINT(DISCOPT):   run;

NOTE: There were 10 observations read from the data set WORK.L.
NOTE: The data set WORK.L has 10 observations and 2 variables.
NOTE: PROCEDURE SORT used:
      real time             0.01 seconds
      cpu time              0.01 seconds


MPRINT(DISCOPT):   data constrai;
MPRINT(DISCOPT):   merge u l;
MPRINT(DISCOPT):   by ticker;
MPRINT(DISCOPT):   prev = -1;
MPRINT(DISCOPT):   run;

NOTE: There were 10 observations read from the data set WORK.U.
NOTE: There were 10 observations read from the data set WORK.L.
NOTE: The data set WORK.CONSTRAI has 10 observations and 4 variables.
NOTE: DATA statement used:
      real time             0.01 seconds
      cpu time              0.01 seconds


MPRINT(PUSHONE):   data _const;
MPRINT(PUSHONE):   length ticker $8;
MPRINT(PUSHONE):   set constrai;
MPRINT(PUSHONE):   pos = 1;
MPRINT(PUSHONE):   prev = -1;

NOTE: There were 10 observations read from the data set WORK.CONSTRAI.
NOTE: The data set WORK._CONST has 10 observations and 5 variables.
NOTE: DATA statement used:
      real time             0.01 seconds
      cpu time              0.00 seconds


MPRINT(PUSHONE):   proc append base=stack data=_const;
NOTE: Appending WORK._CONST to WORK.STACK.
NOTE: There were 10 observations read from the data set WORK._CONST.
NOTE: 10 observations added.
NOTE: The data set WORK.STACK has 10 observations and 5 variables.
NOTE: PROCEDURE APPEND used:
```

```
      real time           0.01 seconds
      cpu time            0.01 seconds


MPRINT(PUSHONE):   data _consto;
MPRINT(PUSHONE):   format ticker $8.;
MPRINT(PUSHONE):   pos = 1;
MPRINT(PUSHONE):   ticker = "";
MPRINT(PUSHONE):   cur = -1;
MPRINT(PUSHONE):   prev = -1;
MPRINT(PUSHONE):   opt = 0;
MPRINT(PUSHONE):   objval = -99999;
MPRINT(PUSHONE):   dist2min = 999;
MPRINT(PUSHONE):   run;

NOTE: The data set WORK._CONSTO has 1 observations and 7 variables.
NOTE: DATA statement used:
      real time           0.01 seconds
      cpu time            0.01 seconds


MPRINT(PUSHONE):   proc append base=stacko data=_consto;
MPRINT(DISCOPT):   ;

NOTE: Appending WORK._CONSTO to WORK.STACKO.
NOTE: There were 1 observations read from the data set WORK._CONSTO.
NOTE: 1 observations added.
NOTE: The data set WORK.STACKO has 1 observations and 7 variables.
NOTE: PROCEDURE APPEND used:
      real time           0.01 seconds
      cpu time            0.02 seconds


MPRINT(POPSTK):   proc sql noprint;
MPRINT(POPSTK):   create table _conpos as select so.objval, so.dist2min, so.opt, so.pos, count(*) as cnt from stack s, stacko so
where s.pos=so.pos and s.upper ne 0 and s.lower ne 0.05 group by so.dist2min, so.objval, so.opt, so.pos order by so.dist2min,
so.objval, so.opt, so.pos;
NOTE: Table WORK._CONPOS created, with 1 rows and 5 columns.

NOTE: PROCEDURE SQL used:
      real time           0.02 seconds
      cpu time            0.01 seconds


MPRINT(POPSTK):   proc sql;
MPRINT(POPSTK):   select count(*) into :cntcnt from _conpos;
NOTE: The PROCEDURE SQL printed page 12.
NOTE: PROCEDURE SQL used:
      real time           0.00 seconds
      cpu time            0.01 seconds


MPRINT(POPSTK):   data _null_;
MPRINT(POPSTK):   set _conpos(obs=1);
MPRINT(POPSTK):   call symput("minpos",trim(left(put(pos,8.))));
MPRINT(POPSTK):   run;

NOTE: There were 1 observations read from the data set WORK._CONPOS.
```

```
NOTE: DATA statement used:
      real time           0.00 seconds
      cpu time            0.00 seconds


MPRINT(POPSTK):   data constrai stack;
MPRINT(POPSTK):   set stack;
MPRINT(POPSTK):   if pos = 1 then do;
MPRINT(POPSTK):   output constrai;
MPRINT(POPSTK):   end;
MPRINT(POPSTK):   else do;
MPRINT(POPSTK):   if pos > 1 then pos = pos - 1;
MPRINT(POPSTK):   output stack;
MPRINT(POPSTK):   end;
MPRINT(POPSTK):   run;

NOTE: There were 10 observations read from the data set WORK.STACK.
NOTE: The data set WORK.CONSTRAI has 10 observations and 5 variables.
NOTE: The data set WORK.STACK has 0 observations and 5 variables.
NOTE: DATA statement used:
      real time           0.03 seconds
      cpu time            0.00 seconds


MPRINT(POPSTK):   data stacko;
MPRINT(POPSTK):   set stacko;
MPRINT(POPSTK):   if pos = 1 then do;
MPRINT(POPSTK):   call symput("sobjval",objval);
MPRINT(POPSTK):   put ".popping " ticker ":(" objval "at " prev ") -> " cur "out of 1" ;
MPRINT(POPSTK):   end;
MPRINT(POPSTK):   else do;
MPRINT(POPSTK):   if pos > 1 then pos = pos - 1;
MPRINT(POPSTK):   output;
MPRINT(POPSTK):   end;
MPRINT(POPSTK):   run;

NOTE: Numeric values have been converted to character values at the places given by: (Line):(Column).
      11788:180
.popping   :(-99999 at -1 ) -> -1 out of 1
NOTE: There were 1 observations read from the data set WORK.STACKO.
NOTE: The data set WORK.STACKO has 0 observations and 7 variables.
NOTE: DATA statement used:
      real time           0.01 seconds
      cpu time            0.01 seconds


MPRINT(DISCOPT):  ;
MPRINT(DISCOPT):  data _null_;
MPRINT(DISCOPT):  title "Iteration 1 minobjfn = 99999 last objfnval =  ";
MPRINT(DISCOPT):  run;

NOTE: DATA statement used:
      real time           0.00 seconds
      cpu time            0.00 seconds


MPRINT(RUNCOPT):  proc transpose data=constrai(keep=ticker lower) out=l;
MPRINT(RUNCOPT):  id ticker;
```

```
NOTE: There were 10 observations read from the data set WORK.CONSTRAI.
NOTE: The data set WORK.L has 1 observations and 11 variables.
NOTE: PROCEDURE TRANSPOSE used:
      real time             0.01 seconds
      cpu time              0.01 seconds


MPRINT(RUNCOPT):   proc transpose data=constrai(keep=ticker upper) out=u;
MPRINT(RUNCOPT):   id ticker;

NOTE: There were 10 observations read from the data set WORK.CONSTRAI.
NOTE: The data set WORK.U has 1 observations and 11 variables.
NOTE: PROCEDURE TRANSPOSE used:
      real time             0.01 seconds
      cpu time              0.00 seconds


MPRINT(RUNCOPT):   data l;
MPRINT(RUNCOPT):   set l;
MPRINT(RUNCOPT):   _name_ = "LOWER";
MPRINT(RUNCOPT):   _type_ = "lowerbd";

NOTE: There were 1 observations read from the data set WORK.L.
NOTE: The data set WORK.L has 1 observations and 12 variables.
NOTE: DATA statement used:
      real time             0.01 seconds
      cpu time              0.00 seconds


MPRINT(RUNCOPT):   data u;
MPRINT(RUNCOPT):   set u;
MPRINT(RUNCOPT):   _name_ = "UPPER";
MPRINT(RUNCOPT):   _type_ = "upperbd";
MPRINT(RUNCOPT):   run;

NOTE: There were 1 observations read from the data set WORK.U.
NOTE: The data set WORK.U has 1 observations and 12 variables.
NOTE: DATA statement used:
      real time             0.01 seconds
      cpu time              0.00 seconds


MPRINT(RUNCOPT):   data pf.inquad;
MPRINT(RUNCOPT):   set pf.inquad(where=(_name_ not in ("LOWER","UPPER"))) l u;
MPRINT(RUNCOPT):   run;

NOTE: There were 24 observations read from the data set PF.INQUAD.
      WHERE _name_ not in ('LOWER', 'UPPER');
NOTE: There were 1 observations read from the data set WORK.L.
NOTE: There were 1 observations read from the data set WORK.U.
NOTE: The data set PF.INQUAD has 26 observations and 12 variables.
NOTE: DATA statement used:
      real time             0.02 seconds
      cpu time              0.02 seconds


.quad call 1 .
```

```
MPRINT(CALLNLP):   proc nlp inquad=pf.inquad outest=toutquad cov=2 ;
MPRINT(CALLNLP):   min;
MPRINT(CALLNLP):   parms CPO FBNDX FCNTX FGRIX FLPSX FMAGX FOSFX FUSEX _GIC_ ;
MPRINT(CALLNLP):   run;

NOTE: Your code contains 0 program statements.
NOTE: Analytic derivatives are used with a quadratic optimization problem.
WARNING: A total of 1 identical linear constraints are deleted.
NOTE: Initial value of parameter CPO is set randomly to 0.0374011193.
NOTE: Initial value of parameter FBNDX is set randomly to 0.692039742.
NOTE: Initial value of parameter FCNTX is set randomly to 0.7295711342.
NOTE: Initial value of parameter FGRIX is set randomly to 0.3651817298.
NOTE: Initial value of parameter FLPSX is set randomly to 0.143754704.
NOTE: Initial value of parameter FMAGX is set randomly to 0.9569836841.
NOTE: Initial value of parameter FOSFX is set randomly to 0.2063702742.
NOTE: Initial value of parameter FUSEX is set randomly to 0.7789351362.
NOTE: Initial value of parameter _GIC_ is set randomly to 0.0473946906.
NOTE: Initial point was changed to be feasible for boundary and linear constraints.
NOTE: ABSGCONV convergence criterion satisfied.
NOTE: There were 26 observations read from the data set PF.INQUAD.
NOTE: The data set WORK.TOUTQUAD has 63 observations and 14 variables.
NOTE: The PROCEDURE NLP printed pages 13-16.
NOTE: PROCEDURE NLP used:
      real time           0.13 seconds
      cpu time            0.06 seconds


MPRINT(CALLNLP):   proc sql noprint;
MPRINT(CALLNLP):   select _rhs_ into :objfnval from toutquad where _type_ = 'PARMS';
MPRINT(RUNCOPT):   ;
NOTE: PROCEDURE SQL used:
      real time           0.00 seconds
      cpu time            0.00 seconds


MPRINT(Q2CTW):   proc sql;
MPRINT(Q2CTW):   select _rhs_ into :objfnval from toutquad where _type_ = 'PARMS';
NOTE: The PROCEDURE SQL printed page 17.
NOTE: PROCEDURE SQL used:
      real time           0.00 seconds
      cpu time            0.01 seconds


MPRINT(Q2CTW):   data _qctw(drop=_type_ _rhs_);
MPRINT(Q2CTW):   format cticker $8.;
MPRINT(Q2CTW):   set toutquad(drop= _name_ _iter_ _tech_ where=(_type_='PARMS'));
MPRINT(Q2CTW):   cticker = "PARMS";
MPRINT(Q2CTW):   call symput("OBJFNVAL",put(_rhs_,best.));
MPRINT(Q2CTW):   if _rhs_ ne . then do;
MPRINT(Q2CTW):   output;
MPRINT(Q2CTW):   call symput("OSTATUS","0");
MPRINT(Q2CTW):   end;
MPRINT(Q2CTW):   else do;
MPRINT(Q2CTW):   call symput("OSTATUS","1");
MPRINT(Q2CTW):   end;
MPRINT(Q2CTW):   run;

NOTE: There were 1 observations read from the data set WORK.TOUTQUAD.
```

```
      WHERE _type_='PARMS';
NOTE: The data set WORK._QCTW has 1 observations and 10 variables.
NOTE: DATA statement used:
      real time          0.02 seconds
      cpu time           0.01 seconds


MPRINT(Q2CTW):   proc transpose data=_qctw out=_qctw(rename=(_name_=ticker col1=wgt));
MPRINT(Q2CTW):   by cticker;
MPRINT(Q2CTW):   run;

NOTE: There were 1 observations read from the data set WORK._QCTW.
NOTE: The data set WORK._QCTW has 9 observations and 3 variables.
NOTE: PROCEDURE TRANSPOSE used:
      real time          0.01 seconds
      cpu time           0.00 seconds


MPRINT(RUNCOPT):  ;
MPRINT(DIST2MIN):   data _null_;
MPRINT(DIST2MIN):   retain distance 0;
MPRINT(DIST2MIN):   set _qctw end=eof;
MPRINT(DIST2MIN):   if wgt < 0.05 then do;
MPRINT(DIST2MIN):   if wgt < 0.05/2 then distance = distance + wgt;
MPRINT(DIST2MIN):   else distance = distance + 0.05 - wgt;
MPRINT(DIST2MIN):   end;
MPRINT(DIST2MIN):   if eof then call symput("DIST2MIN",put(distance,8.6));
MPRINT(DIST2MIN):   run;

NOTE: There were 9 observations read from the data set WORK._QCTW.
NOTE: DATA statement used:
      real time          0.00 seconds
      cpu time           0.01 seconds


MPRINT(RUNCOPT):  ;
MPRINT(RUNCOPT):   proc sql noprint;
MPRINT(RUNCOPT):   select ticker,wgt into :hitick, :maxlf from _qctw having wgt = (select max(wgt) from _qctw where wgt < 0.05 -
1E-8 and wgt > 1E-8);
MPRINT(RUNCOPT):   select ticker,wgt into :lowtick, :mingz from _qctw having wgt = (select min(wgt) from _qctw where wgt > 1E-8 and
wgt < 0.05 - 1E-8);
MPRINT(DISCOPT):  ;
.pushing FCNTX :(0.2497684956 at 0.040974 ) -> 0 .
NOTE: PROCEDURE SQL used:
      real time          0.02 seconds
      cpu time           0.02 seconds


MPRINT(PUSHBND):   data _const;
MPRINT(PUSHBND):   set constrai;
MPRINT(PUSHBND):   pos = 1;
MPRINT(PUSHBND):   if ticker = "FCNTX" then do;
MPRINT(PUSHBND):   lower = 0;
MPRINT(PUSHBND):   upper = 0;
MPRINT(PUSHBND):   prev = 0.040974;
MPRINT(PUSHBND):   end;

NOTE: There were 10 observations read from the data set WORK.CONSTRAI.
```

```
NOTE: The data set WORK._CONST has 10 observations and 5 variables.
NOTE: DATA statement used:
      real time           0.01 seconds
      cpu time            0.00 seconds


MPRINT(PUSHBND):   proc append base=stack data= _const;
NOTE: Appending WORK._CONST to WORK.STACK.
NOTE: There were 10 observations read from the data set WORK._CONST.
NOTE: 10 observations added.
NOTE: The data set WORK.STACK has 10 observations and 5 variables.
NOTE: PROCEDURE APPEND used:
      real time           0.01 seconds
      cpu time            0.00 seconds


MPRINT(PUSHBND):   data _consto;
MPRINT(PUSHBND):   format ticker $8.;
MPRINT(PUSHBND):   pos = 1;
MPRINT(PUSHBND):   ticker = "FCNTX";
MPRINT(PUSHBND):   cur = 0;
MPRINT(PUSHBND):   prev = 0.040974;
MPRINT(PUSHBND):   objval = 0.2497684956;
MPRINT(PUSHBND):   opt = abs(prev-cur);
MPRINT(PUSHBND):   dist2min = 0.009026;
MPRINT(PUSHBND):   run;

NOTE: The data set WORK._CONSTO has 1 observations and 7 variables.
NOTE: DATA statement used:
      real time           0.01 seconds
      cpu time            0.01 seconds


MPRINT(PUSHBND):   proc append base=stacko data=_consto;
MPRINT(PUSHSTK):  ;
..pushing FCNTX :(0.2497684956 at 0.040974 ) -> 0.05 .

NOTE: Appending WORK._CONSTO to WORK.STACKO.
NOTE: There were 1 observations read from the data set WORK._CONSTO.
NOTE: 1 observations added.
NOTE: The data set WORK.STACKO has 1 observations and 7 variables.
NOTE: PROCEDURE APPEND used:
      real time           0.01 seconds
      cpu time            0.01 seconds


MPRINT(PUSHBND):   data _const;
MPRINT(PUSHBND):   set constrai;
MPRINT(PUSHBND):   pos = 2;
MPRINT(PUSHBND):   if ticker = "FCNTX" then do;
MPRINT(PUSHBND):   lower = 0.05;
MPRINT(PUSHBND):   upper = upper;
MPRINT(PUSHBND):   prev = 0.040974;
MPRINT(PUSHBND):   end;

NOTE: There were 10 observations read from the data set WORK.CONSTRAI.
NOTE: The data set WORK._CONST has 10 observations and 5 variables.
NOTE: DATA statement used:
```

```
      real time             0.01 seconds
      cpu time              0.02 seconds


MPRINT(PUSHBND):   proc append base=stack data= _const;
NOTE: Appending WORK._CONST to WORK.STACK.
NOTE: There were 10 observations read from the data set WORK._CONST.
NOTE: 10 observations added.
NOTE: The data set WORK.STACK has 20 observations and 5 variables.
NOTE: PROCEDURE APPEND used:
      real time             0.01 seconds
      cpu time              0.00 seconds


MPRINT(PUSHBND):   data _consto;
MPRINT(PUSHBND):   format ticker $8.;
MPRINT(PUSHBND):   pos = 2;
MPRINT(PUSHBND):   ticker = "FCNTX";
MPRINT(PUSHBND):   cur = 0.05;
MPRINT(PUSHBND):   prev = 0.040974;
MPRINT(PUSHBND):   objval = 0.2497684956;
MPRINT(PUSHBND):   opt = abs(prev-cur);
MPRINT(PUSHBND):   dist2min = 0.009026;
MPRINT(PUSHBND):   run;

NOTE: The data set WORK._CONSTO has 1 observations and 7 variables.
NOTE: DATA statement used:
      real time             0.04 seconds
      cpu time              0.01 seconds


MPRINT(PUSHBND):   proc append base=stacko data=_consto;
MPRINT(PUSHSTK):  ;
MPRINT(DISCOPT):  ;

NOTE: Appending WORK._CONSTO to WORK.STACKO.
NOTE: There were 1 observations read from the data set WORK._CONSTO.
NOTE: 1 observations added.
NOTE: The data set WORK.STACKO has 2 observations and 7 variables.
NOTE: PROCEDURE APPEND used:
      real time             0.01 seconds
      cpu time              0.00 seconds


MPRINT(DISCOPT):   proc sql;
MPRINT(DISCOPT):   select min(objval) into :potentl from stacko;
NOTE: The PROCEDURE SQL printed page 18.
NOTE: PROCEDURE SQL used:
      real time             0.00 seconds
      cpu time              0.01 seconds


MPRINT(FORCEMIN):   proc sql;
MPRINT(FORCEMIN):   create table _force as select a.ticker,a.wgt,b.lower,b.upper from _qctw a left join constrai b on
a.ticker=b.ticker order by a.wgt;
NOTE: Table WORK._FORCE created, with 9 rows and 4 columns.

NOTE: PROCEDURE SQL used:
```

```
      real time           0.02 seconds
      cpu time            0.02 seconds


MPRINT(FORCEMIN):   data constrai;
MPRINT(FORCEMIN):   set _force nobs=_nobs;
MPRINT(FORCEMIN):   if 0.05*(_nobs-_n_) > 1 or wgt < 0.05/2 then do;
MPRINT(FORCEMIN):   upper = 0;
MPRINT(FORCEMIN):   lower = 0;
MPRINT(FORCEMIN):   end;
MPRINT(FORCEMIN):   else do;
MPRINT(FORCEMIN):   if lower < 0.05 then lower = 0.05;
MPRINT(FORCEMIN):   if upper eq . then upper = 1;
MPRINT(FORCEMIN):   end;
MPRINT(FORCEMIN):   run;

NOTE: There were 9 observations read from the data set WORK._FORCE.
NOTE: The data set WORK.CONSTRAI has 9 observations and 4 variables.
NOTE: DATA statement used:
      real time           0.01 seconds
      cpu time            0.01 seconds


MPRINT(DISCOPT):  ;
MPRINT(DISCOPT):  proc print data=_force;
MPRINT(DISCOPT):  title "_force";
NOTE: There were 9 observations read from the data set WORK._FORCE.
NOTE: The PROCEDURE PRINT printed page 19.
NOTE: PROCEDURE PRINT used:
      real time           0.00 seconds
      cpu time            0.00 seconds


MPRINT(DISCOPT):  proc print data=constrai;
MPRINT(DISCOPT):  title "constrai";
MPRINT(DISCOPT):  run;

NOTE: There were 9 observations read from the data set WORK.CONSTRAI.
NOTE: The PROCEDURE PRINT printed page 20.
NOTE: PROCEDURE PRINT used:
      real time           0.00 seconds
      cpu time            0.00 seconds


MPRINT(RUNCOPT):  proc transpose data=constrai(keep=ticker lower) out=l;
MPRINT(RUNCOPT):  id ticker;

NOTE: There were 9 observations read from the data set WORK.CONSTRAI.
NOTE: The data set WORK.L has 1 observations and 10 variables.
NOTE: PROCEDURE TRANSPOSE used:
      real time           0.01 seconds
      cpu time            0.00 seconds


MPRINT(RUNCOPT):  proc transpose data=constrai(keep=ticker upper) out=u;
MPRINT(RUNCOPT):  id ticker;

NOTE: There were 9 observations read from the data set WORK.CONSTRAI.
```

```
NOTE: The data set WORK.U has 1 observations and 10 variables.
NOTE: PROCEDURE TRANSPOSE used:
      real time             0.01 seconds
      cpu time              0.01 seconds


MPRINT(RUNCOPT):   data l;
MPRINT(RUNCOPT):   set l;
MPRINT(RUNCOPT):   _name_ = "LOWER";
MPRINT(RUNCOPT):   _type_ = "lowerbd";

NOTE: There were 1 observations read from the data set WORK.L.
NOTE: The data set WORK.L has 1 observations and 11 variables.
NOTE: DATA statement used:
      real time             0.01 seconds
      cpu time              0.00 seconds


MPRINT(RUNCOPT):   data u;
MPRINT(RUNCOPT):   set u;
MPRINT(RUNCOPT):   _name_ = "UPPER";
MPRINT(RUNCOPT):   _type_ = "upperbd";
MPRINT(RUNCOPT):   run;

NOTE: There were 1 observations read from the data set WORK.U.
NOTE: The data set WORK.U has 1 observations and 11 variables.
NOTE: DATA statement used:
      real time             0.01 seconds
      cpu time              0.01 seconds


MPRINT(RUNCOPT):   data pf.inquad;
MPRINT(RUNCOPT):   set pf.inquad(where=(_name_ not in ("LOWER","UPPER"))) l u;
MPRINT(RUNCOPT):   run;

NOTE: There were 24 observations read from the data set PF.INQUAD.
      WHERE _name_ not in ('LOWER', 'UPPER');
NOTE: There were 1 observations read from the data set WORK.L.
NOTE: There were 1 observations read from the data set WORK.U.
NOTE: The data set PF.INQUAD has 26 observations and 12 variables.
NOTE: DATA statement used:
      real time             0.02 seconds
      cpu time              0.01 seconds


...quad call 2 .
MPRINT(CALLNLP):   proc nlp inquad=pf.inquad outest=toutquad cov=2 ;
MPRINT(CALLNLP):   min;
MPRINT(CALLNLP):   parms CPO FBNDX FCNTX FGRIX FLPSX FMAGX FOSFX FUSEX _GIC_ ;
MPRINT(CALLNLP):   run;

NOTE: Your code contains 0 program statements.
NOTE: Analytic derivatives are used with a quadratic optimization problem.
WARNING: A total of 1 identical linear constraints are deleted.
NOTE: Initial value of parameter CPO is set randomly to 0.2223636891.
NOTE: Initial value of parameter FBNDX is set randomly to 0.6621284576.
NOTE: Initial value of parameter FCNTX is set randomly to 0.1293954403.
NOTE: Initial value of parameter FGRIX is set randomly to 0.6245803752.
```

```
NOTE: Initial value of parameter FLPSX is set randomly to 0.0653572819.
NOTE: Initial value of parameter FMAGX is set randomly to 0.9262610338.
NOTE: Initial value of parameter FOSFX is set randomly to 0.7493494473.
NOTE: Initial value of parameter FUSEX is set randomly to 0.310626859.
NOTE: Initial value of parameter _GIC_ is set randomly to 0.0971887168.
NOTE: Initial point was changed to be feasible for boundary and linear constraints.
NOTE: ABSGCONV convergence criterion satisfied.
NOTE: There were 26 observations read from the data set PF.INQUAD.
NOTE: The data set WORK.TOUTQUAD has 62 observations and 14 variables.
NOTE: The PROCEDURE NLP printed pages 21-24.
NOTE: PROCEDURE NLP used:
      real time           0.15 seconds
      cpu time            0.07 seconds


MPRINT(CALLNLP):   proc sql noprint;
MPRINT(CALLNLP):   select _rhs_ into :objfnval from toutquad where _type_ = 'PARMS';
MPRINT(RUNCOPT):   ;
NOTE: PROCEDURE SQL used:
      real time           0.00 seconds
      cpu time            0.00 seconds


MPRINT(Q2CTW):   proc sql;
MPRINT(Q2CTW):   select _rhs_ into :objfnval from toutquad where _type_ = 'PARMS';
NOTE: The PROCEDURE SQL printed page 25.
NOTE: PROCEDURE SQL used:
      real time           0.00 seconds
      cpu time            0.01 seconds


MPRINT(Q2CTW):   data _qctw(drop=_type_ _rhs_);
MPRINT(Q2CTW):   format cticker $8.;
MPRINT(Q2CTW):   set toutquad(drop= _name_ _iter_ _tech_ where=(_type_='PARMS'));
MPRINT(Q2CTW):   cticker = "PARMS";
MPRINT(Q2CTW):   call symput("OBJFNVAL",put(_rhs_,best.));
MPRINT(Q2CTW):   if _rhs_ ne . then do;
MPRINT(Q2CTW):   output;
MPRINT(Q2CTW):   call symput("OSTATUS","0");
MPRINT(Q2CTW):   end;
MPRINT(Q2CTW):   else do;
MPRINT(Q2CTW):   call symput("OSTATUS","1");
MPRINT(Q2CTW):   end;
MPRINT(Q2CTW):   run;

NOTE: There were 1 observations read from the data set WORK.TOUTQUAD.
      WHERE _type_='PARMS';
NOTE: The data set WORK._QCTW has 1 observations and 10 variables.
NOTE: DATA statement used:
      real time           0.02 seconds
      cpu time            0.02 seconds


MPRINT(Q2CTW):   proc transpose data=_qctw out=_qctw(rename=(_name_=ticker col1=wgt));
MPRINT(Q2CTW):   by cticker;
MPRINT(Q2CTW):   run;

NOTE: There were 1 observations read from the data set WORK._QCTW.
```

```
NOTE: The data set WORK._QCTW has 9 observations and 3 variables.
NOTE: PROCEDURE TRANSPOSE used:
      real time             0.01 seconds
      cpu time              0.02 seconds


MPRINT(RUNCOPT):  ;
MPRINT(DIST2MIN):   data _null_;
MPRINT(DIST2MIN):   retain distance 0;
MPRINT(DIST2MIN):   set _qctw end=eof;
MPRINT(DIST2MIN):   if wgt < 0.05 then do;
MPRINT(DIST2MIN):   if wgt < 0.05/2 then distance = distance + wgt;
MPRINT(DIST2MIN):   else distance = distance + 0.05 - wgt;
MPRINT(DIST2MIN):   end;
MPRINT(DIST2MIN):   if eof then call symput("DIST2MIN",put(distance,8.6));
MPRINT(DIST2MIN):   run;

NOTE: There were 9 observations read from the data set WORK._QCTW.
NOTE: DATA statement used:
      real time             0.00 seconds
      cpu time              0.01 seconds


MPRINT(RUNCOPT):  ;
MPRINT(RUNCOPT):   proc sql noprint;
MPRINT(RUNCOPT):   select ticker,wgt into :hitick, :maxlf from _qctw having wgt = (select max(wgt) from _qctw where wgt < 0.05 -
1E-8 and wgt > 1E-8);
NOTE: No rows were selected.
MPRINT(RUNCOPT):   select ticker,wgt into :lowtick, :mingz from _qctw having wgt = (select min(wgt) from _qctw where wgt > 1E-8 and
wgt < 0.05 - 1E-8);
NOTE: No rows were selected.
MPRINT(DISCOPT):  ;
...************************** force low = 0.2499677559 vs potential 0.249768 on quadcall 2 markx*******
NOTE: PROCEDURE SQL used:
      real time             0.01 seconds
      cpu time              0.02 seconds


MPRINT(DISCOPT):   data pf.outquad;
MPRINT(DISCOPT):   set toutquad;
MPRINT(DISCOPT):   run;

NOTE: There were 62 observations read from the data set WORK.TOUTQUAD.
NOTE: The data set PF.OUTQUAD has 62 observations and 14 variables.
NOTE: DATA statement used:
      real time             0.01 seconds
      cpu time              0.02 seconds


MPRINT(POPSTK):   proc sql noprint;
MPRINT(POPSTK):   create table _conpos as select so.objval, so.dist2min, so.opt, so.pos, count(*) as cnt from stack s, stacko so
where s.pos=so.pos and s.upper ne 0 and s.lower ne 0.05 group by so.dist2min, so.objval, so.opt, so.pos order by so.dist2min,
so.objval, so.opt, so.pos;
NOTE: Table WORK._CONPOS created, with 2 rows and 5 columns.

NOTE: PROCEDURE SQL used:
      real time             0.02 seconds
      cpu time              0.02 seconds
```

```
MPRINT(POPSTK):   proc sql;
MPRINT(POPSTK):   select count(*) into :cntcnt from _conpos;
NOTE: The PROCEDURE SQL printed page 26.
NOTE: PROCEDURE SQL used:
      real time           0.00 seconds
      cpu time            0.00 seconds


MPRINT(POPSTK):   data _null_;
MPRINT(POPSTK):   set _conpos(obs=1);
MPRINT(POPSTK):   call symput("minpos",trim(left(put(pos,8.))));
MPRINT(POPSTK):   run;

NOTE: There were 1 observations read from the data set WORK._CONPOS.
NOTE: DATA statement used:
      real time           0.00 seconds
      cpu time            0.00 seconds


MPRINT(POPSTK):   data constrai stack;
MPRINT(POPSTK):   set stack;
MPRINT(POPSTK):   if pos = 2 then do;
MPRINT(POPSTK):   output constrai;
MPRINT(POPSTK):   end;
MPRINT(POPSTK):   else do;
MPRINT(POPSTK):   if pos > 2 then pos = pos - 1;
MPRINT(POPSTK):   output stack;
MPRINT(POPSTK):   end;
MPRINT(POPSTK):   run;

NOTE: There were 20 observations read from the data set WORK.STACK.
NOTE: The data set WORK.CONSTRAI has 10 observations and 5 variables.
NOTE: The data set WORK.STACK has 10 observations and 5 variables.
NOTE: DATA statement used:
      real time           0.02 seconds
      cpu time            0.00 seconds


MPRINT(POPSTK):   data stacko;
MPRINT(POPSTK):   set stacko;
MPRINT(POPSTK):   if pos = 2 then do;
MPRINT(POPSTK):   call symput("sobjval",objval);
MPRINT(POPSTK):   put "..popping " ticker ":(" objval "at " prev ") -> " cur "out of 2" ;
MPRINT(POPSTK):   end;
MPRINT(POPSTK):   else do;
MPRINT(POPSTK):   if pos > 2 then pos = pos - 1;
MPRINT(POPSTK):   output;
MPRINT(POPSTK):   end;
MPRINT(POPSTK):   run;

NOTE: Numeric values have been converted to character values at the places given by: (Line):(Column).
      11788:180
..popping FCNTX :(0.2497684956 at 0.040974 ) -> 0.05 out of 2
NOTE: There were 2 observations read from the data set WORK.STACKO.
NOTE: The data set WORK.STACKO has 1 observations and 7 variables.
NOTE: DATA statement used:
```

```
      real time           0.01 seconds
      cpu time            0.01 seconds


MPRINT(DISCOPT):  ;
MPRINT(DISCOPT):   data _null_;
MPRINT(DISCOPT):   title "Iteration 2 minobjfn = 0.2499677559 last objfnval = 0.2499677559 ";
MPRINT(DISCOPT):   run;

NOTE: DATA statement used:
      real time           0.00 seconds
      cpu time            0.00 seconds


MPRINT(RUNCOPT):   proc transpose data=constrai(keep=ticker lower) out=l;
MPRINT(RUNCOPT):   id ticker;

NOTE: There were 10 observations read from the data set WORK.CONSTRAI.
NOTE: The data set WORK.L has 1 observations and 11 variables.
NOTE: PROCEDURE TRANSPOSE used:
      real time           0.01 seconds
      cpu time            0.00 seconds


MPRINT(RUNCOPT):   proc transpose data=constrai(keep=ticker upper) out=u;
MPRINT(RUNCOPT):   id ticker;

NOTE: There were 10 observations read from the data set WORK.CONSTRAI.
NOTE: The data set WORK.U has 1 observations and 11 variables.
NOTE: PROCEDURE TRANSPOSE used:
      real time           0.01 seconds
      cpu time            0.00 seconds


MPRINT(RUNCOPT):   data l;
MPRINT(RUNCOPT):   set l;
MPRINT(RUNCOPT):   _name_ = "LOWER";
MPRINT(RUNCOPT):   _type_ = "lowerbd";

NOTE: There were 1 observations read from the data set WORK.L.
NOTE: The data set WORK.L has 1 observations and 12 variables.
NOTE: DATA statement used:
      real time           0.01 seconds
      cpu time            0.00 seconds


MPRINT(RUNCOPT):   data u;
MPRINT(RUNCOPT):   set u;
MPRINT(RUNCOPT):   _name_ = "UPPER";
MPRINT(RUNCOPT):   _type_ = "upperbd";
MPRINT(RUNCOPT):   run;

NOTE: There were 1 observations read from the data set WORK.U.
NOTE: The data set WORK.U has 1 observations and 12 variables.
NOTE: DATA statement used:
      real time           0.02 seconds
      cpu time            0.00 seconds
```

```
MPRINT(RUNCOPT):   data pf.inquad;
MPRINT(RUNCOPT):   set pf.inquad(where=(_name_ not in ("LOWER","UPPER"))) l u;
MPRINT(RUNCOPT):   run;

NOTE: There were 24 observations read from the data set PF.INQUAD.
      WHERE _name_ not in ('LOWER', 'UPPER');
NOTE: There were 1 observations read from the data set WORK.L.
NOTE: There were 1 observations read from the data set WORK.U.
NOTE: The data set PF.INQUAD has 26 observations and 12 variables.
NOTE: DATA statement used:
      real time          0.01 seconds
      cpu time           0.02 seconds


..quad call 3 .
MPRINT(CALLNLP):   proc nlp inquad=pf.inquad outest=toutquad cov=2 ;
MPRINT(CALLNLP):   min;
MPRINT(CALLNLP):   parms CPO FBNDX FCNTX FGRIX FLPSX FMAGX FOSFX FUSEX _GIC_ ;
MPRINT(CALLNLP):   run;

NOTE: Your code contains 0 program statements.
NOTE: Analytic derivatives are used with a quadratic optimization problem.
WARNING: A total of 1 identical linear constraints are deleted.
NOTE: Initial value of parameter CPO is set randomly to 0.4073262589.
NOTE: Initial value of parameter FBNDX is set randomly to 0.6322171733.
NOTE: Initial value of parameter FCNTX is set randomly to 0.5292197464.
NOTE: Initial value of parameter FGRIX is set randomly to 0.8839790206.
NOTE: Initial value of parameter FLPSX is set randomly to 0.9869598597.
NOTE: Initial value of parameter FMAGX is set randomly to 0.8955383836.
NOTE: Initial value of parameter FOSFX is set randomly to 0.2923286205.
NOTE: Initial value of parameter FUSEX is set randomly to 0.8423185818.
NOTE: Initial value of parameter _GIC_ is set randomly to 0.146982743.
NOTE: Initial point was changed to be feasible for boundary and linear constraints.
NOTE: ABSGCONV convergence criterion satisfied.
NOTE: There were 26 observations read from the data set PF.INQUAD.
NOTE: The data set WORK.TOUTQUAD has 62 observations and 14 variables.
NOTE: The PROCEDURE NLP printed pages 27-30.
NOTE: PROCEDURE NLP used:
      real time          0.12 seconds
      cpu time           0.06 seconds


MPRINT(CALLNLP):   proc sql noprint;
MPRINT(CALLNLP):   select _rhs_ into :objfnval from toutquad where _type_ = 'PARMS';
MPRINT(RUNCOPT):   ;
NOTE: PROCEDURE SQL used:
      real time          0.00 seconds
      cpu time           0.00 seconds


MPRINT(Q2CTW):   proc sql;
MPRINT(Q2CTW):   select _rhs_ into :objfnval from toutquad where _type_ = 'PARMS';
NOTE: The PROCEDURE SQL printed page 31.
NOTE: PROCEDURE SQL used:
      real time          0.00 seconds
      cpu time           0.01 seconds
```

```
MPRINT(Q2CTW):   data _qctw(drop=_type_ _rhs_);
MPRINT(Q2CTW):   format cticker $8.;
MPRINT(Q2CTW):   set toutquad(drop= _name_ _iter_ _tech_ where=(_type_='PARMS'));
MPRINT(Q2CTW):   cticker = "PARMS";
MPRINT(Q2CTW):   call symput("OBJFNVAL",put(_rhs_,best.));
MPRINT(Q2CTW):   if _rhs_ ne . then do;
MPRINT(Q2CTW):   output;
MPRINT(Q2CTW):   call symput("OSTATUS","0");
MPRINT(Q2CTW):   end;
MPRINT(Q2CTW):   else do;
MPRINT(Q2CTW):   call symput("OSTATUS","1");
MPRINT(Q2CTW):   end;
MPRINT(Q2CTW):   run;

NOTE: There were 1 observations read from the data set WORK.TOUTQUAD.
      WHERE _type_='PARMS';
NOTE: The data set WORK._QCTW has 1 observations and 10 variables.
NOTE: DATA statement used:
      real time          0.01 seconds
      cpu time           0.01 seconds


MPRINT(Q2CTW):   proc transpose data=_qctw out=_qctw(rename=(_name_=ticker col1=wgt));
MPRINT(Q2CTW):   by cticker;
MPRINT(Q2CTW):   run;

NOTE: There were 1 observations read from the data set WORK._QCTW.
NOTE: The data set WORK._QCTW has 9 observations and 3 variables.
NOTE: PROCEDURE TRANSPOSE used:
      real time          0.02 seconds
      cpu time           0.01 seconds


MPRINT(RUNCOPT):  ;
MPRINT(DIST2MIN):   data _null_;
MPRINT(DIST2MIN):   retain distance 0;
MPRINT(DIST2MIN):   set _qctw end=eof;
MPRINT(DIST2MIN):   if wgt < 0.05 then do;
MPRINT(DIST2MIN):   if wgt < 0.05/2 then distance = distance + wgt;
MPRINT(DIST2MIN):   else distance = distance + 0.05 - wgt;
MPRINT(DIST2MIN):   end;
MPRINT(DIST2MIN):   if eof then call symput("DIST2MIN",put(distance,8.6));
MPRINT(DIST2MIN):   run;

NOTE: There were 9 observations read from the data set WORK._QCTW.
NOTE: DATA statement used:
      real time          0.00 seconds
      cpu time           0.01 seconds


MPRINT(RUNCOPT):  ;
MPRINT(RUNCOPT):   proc sql noprint;
MPRINT(RUNCOPT):   select ticker,wgt into :hitick, :maxlf from _qctw having wgt = (select max(wgt) from _qctw where wgt < 0.05 -
1E-8 and wgt > 1E-8);
NOTE: No rows were selected.
MPRINT(RUNCOPT):   select ticker,wgt into :lowtick, :mingz from _qctw having wgt = (select min(wgt) from _qctw where wgt > 1E-8 and
wgt < 0.05 - 1E-8);
```

```
NOTE: No rows were selected.
MPRINT(DISCOPT):  ;
NOTE: PROCEDURE SQL used:
      real time           0.02 seconds
      cpu time            0.02 seconds


MPRINT(POPSTK):   proc sql noprint;
MPRINT(POPSTK):   create table _conpos as select so.objval, so.dist2min, so.opt, so.pos, count(*) as cnt from stack s, stacko so
where s.pos=so.pos and s.upper ne 0 and s.lower ne 0.05 group by so.dist2min, so.objval, so.opt, so.pos order by so.dist2min,
so.objval, so.opt, so.pos;
NOTE: Table WORK._CONPOS created, with 1 rows and 5 columns.

NOTE: PROCEDURE SQL used:
      real time           0.02 seconds
      cpu time            0.01 seconds


MPRINT(POPSTK):   proc sql;
MPRINT(POPSTK):   select count(*) into :cntcnt from _conpos;
NOTE: The PROCEDURE SQL printed page 32.
NOTE: PROCEDURE SQL used:
      real time           0.00 seconds
      cpu time            0.01 seconds


MPRINT(POPSTK):   data _null_;
MPRINT(POPSTK):   set _conpos(obs=1);
MPRINT(POPSTK):   call symput("minpos",trim(left(put(pos,8.))));
MPRINT(POPSTK):   run;

NOTE: There were 1 observations read from the data set WORK._CONPOS.
NOTE: DATA statement used:
      real time           0.00 seconds
      cpu time            0.01 seconds


MPRINT(POPSTK):   data constrai stack;
MPRINT(POPSTK):   set stack;
MPRINT(POPSTK):   if pos = 1 then do;
MPRINT(POPSTK):   output constrai;
MPRINT(POPSTK):   end;
MPRINT(POPSTK):   else do;
MPRINT(POPSTK):   if pos > 1 then pos = pos - 1;
MPRINT(POPSTK):   output stack;
MPRINT(POPSTK):   end;
MPRINT(POPSTK):   run;

NOTE: There were 10 observations read from the data set WORK.STACK.
NOTE: The data set WORK.CONSTRAI has 10 observations and 5 variables.
NOTE: The data set WORK.STACK has 0 observations and 5 variables.
NOTE: DATA statement used:
      real time           0.03 seconds
      cpu time            0.02 seconds


MPRINT(POPSTK):   data stacko;
MPRINT(POPSTK):   set stacko;
```

```
MPRINT(POPSTK):   if pos = 1 then do;
MPRINT(POPSTK):   call symput("sobjval",objval);
MPRINT(POPSTK):   put ".popping " ticker ":(" objval "at " prev ") -> " cur "out of 1" ;
MPRINT(POPSTK):   end;
MPRINT(POPSTK):   else do;
MPRINT(POPSTK):   if pos > 1 then pos = pos - 1;
MPRINT(POPSTK):   output;
MPRINT(POPSTK):   end;
MPRINT(POPSTK):   run;

NOTE: Numeric values have been converted to character values at the places given by: (Line):(Column).
      11788:180
.popping FCNTX :(0.2497684956 at 0.040974 ) -> 0 out of 1
NOTE: There were 1 observations read from the data set WORK.STACKO.
NOTE: The data set WORK.STACKO has 0 observations and 7 variables.
NOTE: DATA statement used:
      real time          0.01 seconds
      cpu time           0.01 seconds


MPRINT(DISCOPT):  ;
MPRINT(DISCOPT):  data _null_;
MPRINT(DISCOPT):  title "Iteration 3 minobjfn = 0.2499677559 last objfnval = 0.2499677559 ";
MPRINT(DISCOPT):  run;

NOTE: DATA statement used:
      real time          0.00 seconds
      cpu time           0.00 seconds


MPRINT(RUNCOPT):  proc transpose data=constrai(keep=ticker lower) out=l;
MPRINT(RUNCOPT):  id ticker;

NOTE: There were 10 observations read from the data set WORK.CONSTRAI.
NOTE: The data set WORK.L has 1 observations and 11 variables.
NOTE: PROCEDURE TRANSPOSE used:
      real time          0.01 seconds
      cpu time           0.01 seconds


MPRINT(RUNCOPT):  proc transpose data=constrai(keep=ticker upper) out=u;
MPRINT(RUNCOPT):  id ticker;

NOTE: There were 10 observations read from the data set WORK.CONSTRAI.
NOTE: The data set WORK.U has 1 observations and 11 variables.
NOTE: PROCEDURE TRANSPOSE used:
      real time          0.01 seconds
      cpu time           0.00 seconds


MPRINT(RUNCOPT):  data l;
MPRINT(RUNCOPT):  set l;
MPRINT(RUNCOPT):  _name_ = "LOWER";
MPRINT(RUNCOPT):  _type_ ="lowerbd";

NOTE: There were 1 observations read from the data set WORK.L.
NOTE: The data set WORK.L has 1 observations and 12 variables.
NOTE: DATA statement used:
```

```
      real time           0.01 seconds
      cpu time            0.02 seconds


MPRINT(RUNCOPT):   data u;
MPRINT(RUNCOPT):   set u;
MPRINT(RUNCOPT):   _name_ = "UPPER";
MPRINT(RUNCOPT):   _type_ = "upperbd";
MPRINT(RUNCOPT):   run;

NOTE: There were 1 observations read from the data set WORK.U.
NOTE: The data set WORK.U has 1 observations and 12 variables.
NOTE: DATA statement used:
      real time           0.01 seconds
      cpu time            0.02 seconds


MPRINT(RUNCOPT):   data pf.inquad;
MPRINT(RUNCOPT):   set pf.inquad(where=(_name_ not in ("LOWER","UPPER"))) l u;
MPRINT(RUNCOPT):   run;

NOTE: There were 24 observations read from the data set PF.INQUAD.
      WHERE _name_ not in ('LOWER', 'UPPER');
NOTE: There were 1 observations read from the data set WORK.L.
NOTE: There were 1 observations read from the data set WORK.U.
NOTE: The data set PF.INQUAD has 26 observations and 12 variables.
NOTE: DATA statement used:
      real time           0.02 seconds
      cpu time            0.03 seconds


.quad call 4 .
MPRINT(CALLNLP):   proc nlp inquad=pf.inquad outest=toutquad cov=2 ;
MPRINT(CALLNLP):   min;
MPRINT(CALLNLP):   parms CPO FBNDX FCNTX FGRIX FLPSX FMAGX FOSFX FUSEX _GIC_ ;
MPRINT(CALLNLP):   run;

NOTE: Your code contains 0 program statements.
NOTE: Analytic derivatives are used with a quadratic optimization problem.
WARNING: A total of 1 identical linear constraints are deleted.
NOTE: Initial value of parameter CPO is set randomly to 0.4073262589.
NOTE: Initial value of parameter FBNDX is set randomly to 0.6322171733.
NOTE: Initial value of parameter FCNTX is set randomly to 0.5292197464.
NOTE: Initial value of parameter FGRIX is set randomly to 0.8839790206.
NOTE: Initial value of parameter FLPSX is set randomly to 0.9869598597.
NOTE: Initial value of parameter FMAGX is set randomly to 0.8955383836.
NOTE: Initial value of parameter FOSFX is set randomly to 0.2923286205.
NOTE: Initial value of parameter FUSEX is set randomly to 0.8423185818.
NOTE: Initial value of parameter _GIC_ is set randomly to 0.146982743.
NOTE: Initial point was changed to be feasible for boundary and linear constraints.
NOTE: ABSGCONV convergence criterion satisfied.
NOTE: There were 26 observations read from the data set PF.INQUAD.
NOTE: The data set WORK.TOUTQUAD has 62 observations and 14 variables.
NOTE: The PROCEDURE NLP printed pages 33-36.
NOTE: PROCEDURE NLP used:
      real time           0.14 seconds
      cpu time            0.06 seconds
```

```
MPRINT(CALLNLP):   proc sql noprint;
MPRINT(CALLNLP):   select _rhs_ into :objfnval from toutquad where _type_ = 'PARMS';
MPRINT(RUNCOPT):   ;
NOTE: PROCEDURE SQL used:
      real time           0.00 seconds
      cpu time            0.01 seconds


MPRINT(Q2CTW):   proc sql;
MPRINT(Q2CTW):   select _rhs_ into :objfnval from toutquad where _type_ = 'PARMS';
NOTE: The PROCEDURE SQL printed page 37.
NOTE: PROCEDURE SQL used:
      real time           0.00 seconds
      cpu time            0.01 seconds


MPRINT(Q2CTW):   data _qctw(drop=_type_ _rhs_);
MPRINT(Q2CTW):   format cticker $8.;
MPRINT(Q2CTW):   set toutquad(drop= _name_ _iter_ _tech_ where=(_type_='PARMS'));
MPRINT(Q2CTW):   cticker = "PARMS";
MPRINT(Q2CTW):   call symput("OBJFNVAL",put(_rhs_,best.));
MPRINT(Q2CTW):   if _rhs_ ne . then do;
MPRINT(Q2CTW):   output;
MPRINT(Q2CTW):   call symput("OSTATUS","0");
MPRINT(Q2CTW):   end;
MPRINT(Q2CTW):   else do;
MPRINT(Q2CTW):   call symput("OSTATUS","1");
MPRINT(Q2CTW):   end;
MPRINT(Q2CTW):   run;

NOTE: There were 1 observations read from the data set WORK.TOUTQUAD.
      WHERE _type_='PARMS';
NOTE: The data set WORK._QCTW has 1 observations and 10 variables.
NOTE: DATA statement used:
      real time           0.02 seconds
      cpu time            0.02 seconds


MPRINT(Q2CTW):   proc transpose data=_qctw out=_qctw(rename=(_name_=ticker col1=wgt));
MPRINT(Q2CTW):   by cticker;
MPRINT(Q2CTW):   run;

NOTE: There were 1 observations read from the data set WORK._QCTW.
NOTE: The data set WORK._QCTW has 9 observations and 3 variables.
NOTE: PROCEDURE TRANSPOSE used:
      real time           0.02 seconds
      cpu time            0.01 seconds


MPRINT(RUNCOPT):  ;
MPRINT(DIST2MIN):   data _null_;
MPRINT(DIST2MIN):   retain distance 0;
MPRINT(DIST2MIN):   set _qctw end=eof;
MPRINT(DIST2MIN):   if wgt < 0.05 then do;
MPRINT(DIST2MIN):   if wgt < 0.05/2 then distance = distance + wgt;
MPRINT(DIST2MIN):   else distance = distance + 0.05 - wgt;
MPRINT(DIST2MIN):   end;
```

```
MPRINT(DIST2MIN):   if eof then call symput("DIST2MIN",put(distance,8.6));
MPRINT(DIST2MIN):   run;

NOTE: There were 9 observations read from the data set WORK._QCTW.
NOTE: DATA statement used:
      real time          0.00 seconds
      cpu time           0.01 seconds


MPRINT(RUNCOPT):  ;
MPRINT(RUNCOPT):   proc sql noprint;
MPRINT(RUNCOPT):   select ticker,wgt into :hitick, :maxlf from _qctw having wgt = (select max(wgt) from _qctw where wgt < 0.05 -
1E-8 and wgt > 1E-8);
NOTE: No rows were selected.
MPRINT(RUNCOPT):   select ticker,wgt into :lowtick, :mingz from _qctw having wgt = (select min(wgt) from _qctw where wgt > 1E-8 and
wgt < 0.05 - 1E-8);
NOTE: No rows were selected.
MPRINT(DISCOPT):  ;
NOTE: PROCEDURE SQL used:
      real time          0.01 seconds
      cpu time           0.02 seconds


MPRINT(DISCOPT):   data toutquad;
MPRINT(DISCOPT):   set pf.outquad;
MPRINT(DISCOPT):   run;

NOTE: There were 62 observations read from the data set PF.OUTQUAD.
NOTE: The data set WORK.TOUTQUAD has 62 observations and 14 variables.
NOTE: DATA statement used:
      real time          0.01 seconds
      cpu time           0.02 seconds


Finished after 4 calls to quad, obj=0.2499677559 markx
MPRINT(OPTO_CTW):  ;
MPRINT(Q2CTW):   proc sql;
MPRINT(Q2CTW):   select _rhs_ into :objfnval from pf.outquad where _type_ = 'PARMS';
NOTE: The PROCEDURE SQL printed page 38.
NOTE: PROCEDURE SQL used:
      real time          0.00 seconds
      cpu time           0.01 seconds


MPRINT(Q2CTW):   data _out_ctw(drop=_type_ _rhs_);
MPRINT(Q2CTW):   format cticker $8.;
MPRINT(Q2CTW):   set pf.outquad(drop= _name_ _iter_ _tech_ where=(_type_='PARMS'));
MPRINT(Q2CTW):   cticker = "PF000021";
MPRINT(Q2CTW):   call symput("OBJFNVAL",put(_rhs_,best.));
MPRINT(Q2CTW):   if _rhs_ ne . then do;
MPRINT(Q2CTW):   output;
MPRINT(Q2CTW):   call symput("OSTATUS","0");
MPRINT(Q2CTW):   end;
MPRINT(Q2CTW):   else do;
MPRINT(Q2CTW):   call symput("OSTATUS","1");
MPRINT(Q2CTW):   end;
MPRINT(Q2CTW):   run;
```

```
NOTE: There were 1 observations read from the data set PF.OUTQUAD.
      WHERE _type_='PARMS';
NOTE: The data set WORK._OUT_CTW has 1 observations and 10 variables.
NOTE: DATA statement used:
      real time           0.02 seconds
      cpu time            0.01 seconds


MPRINT(Q2CTW):   proc transpose data=_out_ctw out=_out_ctw(rename=(_name_=ticker col1=wgt));
MPRINT(Q2CTW):   by cticker;
MPRINT(Q2CTW):   run;

NOTE: There were 1 observations read from the data set WORK._OUT_CTW.
NOTE: The data set WORK._OUT_CTW has 9 observations and 3 variables.
NOTE: PROCEDURE TRANSPOSE used:
      real time           0.01 seconds
      cpu time            0.00 seconds


MPRINT(OPTO_CTW):  ;
MPRINT(OPTO_CTW):  data pf.ctw;
MPRINT(OPTO_CTW):  length cticker ticker $8;
MPRINT(OPTO_CTW):  set _out_ctw;
MPRINT(OPTO_CTW):  run;

NOTE: There were 9 observations read from the data set WORK._OUT_CTW.
NOTE: The data set PF.CTW has 9 observations and 3 variables.
NOTE: DATA statement used:
      real time           0.02 seconds
      cpu time            0.01 seconds


MPRINT(OPTO_PF):  ;
MPRINT(CTW2WHOLE_PCT):  data finals;
MPRINT(CTW2WHOLE_PCT):  set pf.ctw(drop=cticker);
MPRINT(CTW2WHOLE_PCT):  wgt=round(wgt,.0001);
MPRINT(CTW2WHOLE_PCT):  run;

NOTE: There were 9 observations read from the data set PF.CTW.
NOTE: The data set WORK.FINALS has 9 observations and 2 variables.
NOTE: DATA statement used:
      real time           0.01 seconds
      cpu time            0.00 seconds


MPRINT(CTW2WHOLE_PCT):  proc sort data=finals;
MPRINT(CTW2WHOLE_PCT):  by ticker;

NOTE: There were 9 observations read from the data set WORK.FINALS.
NOTE: The data set WORK.FINALS has 9 observations and 2 variables.
NOTE: PROCEDURE SORT used:
      real time           0.01 seconds
      cpu time            0.00 seconds


MPRINT(CTW2WHOLE_PCT):  data aret;
MPRINT(CTW2WHOLE_PCT):  set pf.mdl_dtt;
MPRINT(CTW2WHOLE_PCT):  run;
```

```
NOTE: There were 36 observations read from the data set PF.MDL_DTT.
NOTE: The data set WORK.ARET has 36 observations and 12 variables.
NOTE: DATA statement used:
      real time          0.01 seconds
      cpu time           0.01 seconds


MPRINT(W2LTRAN):   proc transpose data=pf.mdl_dtt out=_c2dtr(keep = date ticker col1 rename=(col1=return)) name=ticker;
MPRINT(W2LTRAN):   by date;
MPRINT(W2LTRAN):   run;

NOTE: There were 36 observations read from the data set PF.MDL_DTT.
NOTE: The data set WORK._C2DTR has 396 observations and 3 variables.
NOTE: PROCEDURE TRANSPOSE used:
      real time          0.02 seconds
      cpu time           0.01 seconds


MPRINT(W2LTRAN):   data _c2dtr;
MPRINT(W2LTRAN):   length ticker $8;
MPRINT(W2LTRAN):   set _c2dtr(where=(return ne .));
MPRINT(W2LTRAN):   run;

NOTE: There were 396 observations read from the data set WORK._C2DTR.
      WHERE return not = .;
NOTE: The data set WORK._C2DTR has 396 observations and 3 variables.
NOTE: DATA statement used:
      real time          0.01 seconds
      cpu time           0.02 seconds


MPRINT(DTT2DTR):  ;
MPRINT(CTW2WHOLE_PCT):  ;
MPRINT(CTW2WHOLE_PCT):   proc sql noprint;
MPRINT(CTW2WHOLE_PCT):   create table preset as select a.ticker,a.wgt*100 as wgt, std(return) as std from finals a, _c2dtr b where
a.ticker=b.ticker group by a.ticker,wgt order by ticker;
NOTE: Table WORK.PRESET created, with 9 rows and 3 columns.

NOTE: PROCEDURE SQL used:
      real time          0.03 seconds
      cpu time           0.02 seconds


MPRINT(CTW2WHOLE_PCT):   data preset;
MPRINT(CTW2WHOLE_PCT):   set preset;
MPRINT(CTW2WHOLE_PCT):   if ( wgt ne int(wgt) ) then do;
MPRINT(CTW2WHOLE_PCT):   lowerbd = int(wgt);
MPRINT(CTW2WHOLE_PCT):   upperbd = int(wgt) + 1 ;
MPRINT(CTW2WHOLE_PCT):   end;
MPRINT(CTW2WHOLE_PCT):   else do;
MPRINT(CTW2WHOLE_PCT):   lowerbd = wgt;
MPRINT(CTW2WHOLE_PCT):   upperbd = wgt;
MPRINT(CTW2WHOLE_PCT):   end;
MPRINT(CTW2WHOLE_PCT):   integer=1;
MPRINT(CTW2WHOLE_PCT):   wstd= std*wgt;
MPRINT(CTW2WHOLE_PCT):   run;
```

```
NOTE: There were 9 observations read from the data set WQRK.PRESET.
NOTE: The data set WORK.PRESET has 9 observations and 7 variables.
NOTE: DATA statement used:
      real time             0.02 seconds
      cpu time              0.01 seconds


MPRINT(CTW2WHOLE_PCT):   proc means data=preSet noprint;
MPRINT(CTW2WHOLE_PCT):   var wstd;
MPRINT(CTW2WHOLE_PCT):   output out=_xsum sum=std;
NOTE: There were 9 observations read from the data set WORK.PRESET.
NOTE: The data set WORK._XSUM has 1 observations and 3 variables.
NOTE: PROCEDURE MEANS used:
      real time             0.03 seconds
      cpu time              0.02 seconds


MPRINT(CTW2WHOLE_PCT):   data _xsum;
MPRINT(CTW2WHOLE_PCT):   length ticker $10;
MPRINT(CTW2WHOLE_PCT):   set _xsum;
MPRINT(CTW2WHOLE_PCT):   ticker = "SUM";
MPRINT(CTW2WHOLE_PCT):   wgt = -1;
MPRINT(CTW2WHOLE_PCT):   lowerbd=-1;
MPRINT(CTW2WHOLE_PCT):   upperbd=-1;
MPRINT(CTW2WHOLE_PCT):   integer=1;
MPRINT(CTW2WHOLE_PCT):   wstd=.;

NOTE: There were 1 observations read from the data set WORK._XSUM.
NOTE: The data set WORK._XSUM has 1 observations and 9 variables.
NOTE: DATA statement used:
      real time             0.01 seconds
      cpu time              0.00 seconds


MPRINT(CTW2WHOLE_PCT):   data preset;
MPRINT(CTW2WHOLE_PCT):   set preset _xsum;
MPRINT(CTW2WHOLE_PCT):   run;

NOTE: There were 9 observations read from the data set WORK.PRESET.
NOTE: There were 1 observations read from the data set WORK._XSUM.
NOTE: The data set WORK.PRESET has 10 observations and 9 variables.
NOTE: DATA statement used:
      real time             0.01 seconds
      cpu time              0.00 seconds


MPRINT(CTW2WHOLE_PCT):   proc transpose data=preSet out=inSetx;
MPRINT(CTW2WHOLE_PCT):   id ticker;

NOTE: There were 10 observations read from the data set WORK.PRESET.
NOTE: The data set WORK.INSETX has 8 observations and 11 variables.
NOTE: PROCEDURE TRANSPOSE used:
      real time             0.01 seconds
      cpu time              0.00 seconds


MPRINT(CTW2WHOLE_PCT):   data inSet;
MPRINT(CTW2WHOLE_PCT):   length _type_ $15;
```

```
MPRINT(CTW2WHOLE_PCT):   set insetX(rename=(_name_=_id_) );
MPRINT(CTW2WHOLE_PCT):   if ( _id_ in ("LOWERBD","UPPERBD") ) then do;
MPRINT(CTW2WHOLE_PCT):   _type_ = _id_;
MPRINT(CTW2WHOLE_PCT):   _rhs_ = .;
MPRINT(CTW2WHOLE_PCT):   output;
MPRINT(CTW2WHOLE_PCT):   end;
MPRINT(CTW2WHOLE_PCT):   else if ( _id_ in ("STD") ) then do;
MPRINT(CTW2WHOLE_PCT):   _type_ = "MIN";
MPRINT(CTW2WHOLE_PCT):   _rhs_=.;
MPRINT(CTW2WHOLE_PCT):   output;
MPRINT(CTW2WHOLE_PCT):   _type_ = "GE" ;
MPRINT(CTW2WHOLE_PCT):   _rhs_=0;
MPRINT(CTW2WHOLE_PCT):   output;
MPRINT(CTW2WHOLE_PCT):   end;
MPRINT(CTW2WHOLE_PCT):   else if ( _id_ in ("INTEGER") ) then do;
MPRINT(CTW2WHOLE_PCT):   _type_=_id_ ;
MPRINT(CTW2WHOLE_PCT):   _rhs_=.;
MPRINT(CTW2WHOLE_PCT):   output;
MPRINT(CTW2WHOLE_PCT):   SUM=0;
MPRINT(CTW2WHOLE_PCT):   _type_="EQ";
MPRINT(CTW2WHOLE_PCT):   _rhs_=100;
MPRINT(CTW2WHOLE_PCT):   output;
MPRINT(CTW2WHOLE_PCT):   end;

NOTE: There were 8 observations read from the data set WORK.INSETX.
NOTE: The data set WORK.INSET has 6 observations and 13 variables.
NOTE: DATA statement used:
      real time           0.02 seconds
      cpu time            0.01 seconds


MPRINT(CTW2WHOLE_PCT):   proc lp data=inSet printlevel=-2 primalout=_lpout activeout=actout dualout=duout imaxit=100000 time=25
varselect=far iepsilon=.001 canselect=lifo maxit1=200;
MPRINT(CTW2WHOLE_PCT):   run;

NOTE: The solution in the PRIMALOUT= data set is the optimal integer solution.
NOTE: The solution in the DUALOUT= data set is the optimal integer solution.
NOTE: There were 6 observations read from the data set WORK.INSET.
NOTE: The data set WORK._LPOUT has 13 observations and 10 variables.
NOTE: The data set WORK.ACTOUT has 1 observations and 13 variables.
NOTE: The data set WORK.DUOUT has 3 observations and 9 variables.
NOTE: PROCEDURE LP used:
      real time           0.18 seconds
      cpu time            0.02 seconds


MPRINT(CTW2WHOLE_PCT):   proc sql noprint;
MPRINT(CTW2WHOLE_PCT):   create table pf.mdl_ctw as select a.cticker length=8,a.ticker length=8,b._VALUE_/100 as wgt from pf.ctw a,
_lpout b where a.ticker=b._VAR_ and b._TYPE_ in ("FIXED","INTEGER") and b._VALUE_ ne 0 order by a.cticker,a.ticker;
NOTE: Table PF.MDL_CTW created, with 7 rows and 3 columns.

MPRINT(OPTO_PF):  ;
NOTE: PROCEDURE SQL used:
      real time           0.04 seconds
      cpu time            0.01 seconds


MPRINT(CTWRET):   data _r7;
```

```
MPRINT(CTWRET):   set pf.mdl_ctw(where=(wgt ne 0));
MPRINT(CTWRET):   run;

NOTE: There were 7 observations read from the data set PF.MDL_CTW.
      WHERE wgt not = 0;
NOTE: The data set WORK._R7 has 7 observations and 3 variables.
NOTE: DATA statement used:
      real time          0.01 seconds
      cpu time           0.00 seconds


MPRINT(VARENUM):   proc sql;
MPRINT(VARENUM):   create table _r2 as select distinct ticker as ticker from _r7 order by ticker;
NOTE: Table WORK._R2 created, with 7 rows and 1 columns.

NOTE: PROCEDURE SQL used:
      real time          0.02 seconds
      cpu time           0.01 seconds


MPRINT(VARENUM):   data _r2;
MPRINT(VARENUM):   set _r2;
MPRINT(VARENUM):   format tickid $8.;
MPRINT(VARENUM):   tickid = "_W" || left(trim(put(_n_,8.)));
MPRINT(VARENUM):   run;

NOTE: There were 7 observations read from the data set WORK._R2.
NOTE: The data set WORK._R2 has 7 observations and 2 variables.
NOTE: DATA statement used:
      real time          0.01 seconds
      cpu time           0.01 seconds


MPRINT(CTWRET):   ;
MPRINT(DS2MA):   proc sql noprint;
MPRINT(DS2MA):   select count(distinct ticker) into :_ct0 from _r2 where ticker ne '' ;
NOTE: PROCEDURE SQL used:
      real time          0.02 seconds
      cpu time           0.00 seconds


MPRINT(DS2MA):   proc sql noprint;
MPRINT(DS2MA):   select distinct ticker into :_ct1 - :_ct7 from _r2 where ticker ne '' order by ticker;
MPRINT(CTWRET):   ;
NOTE: PROCEDURE SQL used:
      real time          0.01 seconds
      cpu time           0.01 seconds


MPRINT(V_REPLAC):   proc sort data=_r7 out=_a1;
MPRINT(V_REPLAC):   by ticker;

NOTE: There were 7 observations read from the data set WORK._R7.
NOTE: The data set WORK._A1 has 7 observations and 3 variables.
NOTE: PROCEDURE SORT used:
      real time          0.01 seconds
      cpu time           0.00 seconds
```

```
MPRINT(V_REPLAC):   proc sort data=_r2(keep=ticker tickid) out=_a2;
MPRINT(V_REPLAC):   by ticker;
NOTE: There were 7 observations read from the data set WORK._R2.
NOTE: The data set WORK._A2 has 7 observations and 2 variables.
NOTE: PROCEDURE SORT used:
      real time           0.01 seconds
      cpu time            0.00 seconds


MPRINT(V_REPLAC):   data _r3 (drop=ticker rename=(__x=ticker));
MPRINT(V_REPLAC):   merge _a1(in=ina) _a2(rename=(tickid = __x ticker=ticker));
MPRINT(V_REPLAC):   by ticker;
MPRINT(V_REPLAC):   if ina;
MPRINT(V_REPLAC):   run;

NOTE: There were 7 observations read from the data set WORK._A1.
NOTE: There were 7 observations read from the data set WORK._A2.
NOTE: The data set WORK._R3 has 7 observations and 3 variables.
NOTE: DATA statement used:
      real time           0.02 seconds
      cpu time            0.01 seconds


MPRINT(TRASH):   proc datasets nolist lib=work;
MPRINT(TRASH):   delete _a1 _a2;
MPRINT(TRASH):   run;

NOTE: Deleting WORK._A1 (memtype=DATA).
NOTE: Deleting WORK._A2 (memtype=DATA).
MPRINT(V_REPLAC):   ;
MPRINT(CTWRET):   ;

NOTE: PROCEDURE DATASETS used:
      real time           0.03 seconds
      cpu time            0.01 seconds


MPRINT(L2WTRAN):   proc sort data=_r3;
MPRINT(L2WTRAN):   by cticker;
MPRINT(L2WTRAN):   run;

NOTE: There were 7 observations read from the data set WORK._R3.
NOTE: The data set WORK._R3 has 7 observations and 3 variables.
NOTE: PROCEDURE SORT used:
      real time           0.01 seconds
      cpu time            0.00 seconds


MPRINT(L2WTRAN):   proc transpose data=_r3(keep=wgt ticker cticker) out=_r3(keep=cticker _numeric_);
MPRINT(L2WTRAN):   var wgt;
MPRINT(L2WTRAN):   id ticker;
MPRINT(L2WTRAN):   by cticker;
MPRINT(L2WTRAN):   run;

NOTE: There were 7 observations read from the data set WORK._R3.
NOTE: The data set WORK._R3 has 1 observations and 8 variables.
NOTE: PROCEDURE TRANSPOSE used:
```

```
      real time           0.02 seconds
      cpu time            0.00 seconds


MPRINT(CTWRET):  ;
MPRINT(DSNONULL):   data _r3;
MPRINT(DSNONULL):   set _r3;
MPRINT(DSNONULL):   array _x _numeric_;
MPRINT(DSNONULL):   do over _x;
MPRINT(DSNONULL):   if _x = . then _x = 0;
MPRINT(DSNONULL):   end;
MPRINT(DSNONULL):   run;

NOTE: There were 1 observations read from the data set WORK._R3.
NOTE: The data set WORK._R3 has 1 observations and 8 variables.
NOTE: DATA statement used:
      real time           0.01 seconds
      cpu time            0.01 seconds


MPRINT(CTWRET):  ;
MPRINT(CLEANRET):   proc sql;
MPRINT(CLEANRET):   create table _rc2 as select distinct ticker from _r2;
NOTE: Table WORK._RC2 created, with 7 rows and 1 columns.

NOTE: PROCEDURE SQL used:
      real time           0.01 seconds
      cpu time            0.00 seconds


MPRINT(CLEANRET):   data _rc2;
MPRINT(CLEANRET):   set _rc2 end=eof;
MPRINT(CLEANRET):   if _n_ eq 1 then call symput("_CRA",ticker);
MPRINT(CLEANRET):   if eof then call symput("_CRZ",ticker);
MPRINT(CLEANRET):   return = 0;
MPRINT(CLEANRET):   run;

NOTE: There were 7 observations read from the data set WORK._RC2.
NOTE: The data set WORK._RC2 has 7 observations and 2 variables.
NOTE: DATA statement used:
      real time           0.01 seconds
      cpu time            0.00 seconds


MPRINT(L2WTRAN):   proc transpose data=_rc2(keep=return ticker ) out=_rc2(keep= _numeric_);
MPRINT(L2WTRAN):   var return;
MPRINT(L2WTRAN):   id ticker;
MPRINT(L2WTRAN):  ;
MPRINT(L2WTRAN):   run;

NOTE: There were 7 observations read from the data set WORK._RC2.
NOTE: The data set WORK._RC2 has 1 observations and 7 variables.
NOTE: PROCEDURE TRANSPOSE used:
      real time           0.01 seconds
      cpu time            0.01 seconds


MPRINT(CLEANRET):  ;
```

```
MPRINT(CLEANRET):   data _r5(keep=date CPO -- _GIC_ );
MPRINT(CLEANRET):   merge _rc2 pf.mdl_dtt;
MPRINT(CLEANRET):   output;
MPRINT(CLEANRET):   run;

NOTE: There were 1 observations read from the data set WORK._RC2.
NOTE: There were 36 observations read from the data set PF.MDL_DTT.
NOTE: The data set WORK._R5 has 36 observations and 8 variables.
NOTE: DATA statement used:
      real time          0.02 seconds
      cpu time           0.01 seconds


MPRINT(TRASH):   proc datasets nolist lib=work;
MPRINT(TRASH):   delete _rc2;
MPRINT(TRASH):   run;

NOTE: Deleting WORK._RC2 (memtype=DATA).
MPRINT(CLEANRET):  ;
MPRINT(CTWRET):  ;

NOTE: PROCEDURE DATASETS used:
      real time          0.02 seconds
      cpu time           0.00 seconds


MPRINT(CTWRET):   data _r6;
MPRINT(CTWRET):   set _r5(rename=( CPO=_R1 FCNTX=_R2 FLPSX=_R3 FMAGX=_R4 FOSFX=_R5 FUSEX=_R6 _GIC_=_R7 ));
MPRINT(CTWRET):   run;

NOTE: There were 36 observations read from the data set WORK._R5.
NOTE: The data set WORK._R6 has 36 observations and 8 variables.
NOTE: DATA statement used:
      real time          0.02 seconds
      cpu time           0.01 seconds


MPRINT(CTWRET):   data _r7(keep=cticker date return);
MPRINT(CTWRET):   set _r3 nobs=_nobs;
MPRINT(CTWRET):   array _w[7] _w1 - _w7;
MPRINT(CTWRET):   array _nw[7];
MPRINT(CTWRET):   retain _nw1 - _nw7;
MPRINT(CTWRET):   retain _w1 - _w7 ;
MPRINT(CTWRET):   do _i = 1 to 7;
MPRINT(CTWRET):   _nw[_i] = _w[_i];
MPRINT(CTWRET):   end;
MPRINT(CTWRET):   _zportf + 1;
MPRINT(CTWRET):   if _nobs > 50 and 10*int(_zportf/10) = _zportf then put "portfolio " _zportf " / " _nobs;
MPRINT(CTWRET):   do _rno = 1 to _robs;
MPRINT(CTWRET):   set _r6 nobs=_robs point=_rno;
MPRINT(CTWRET):   array _r[7] _r1 - _r7;
MPRINT(CTWRET):   _sum = 0;
MPRINT(CTWRET):   return = 0;
MPRINT(CTWRET):   do _i = 1 to 7;
MPRINT(CTWRET):   if _nw[_i] ne 0 then do;
MPRINT(CTWRET):   return = return + _nw[_i] * _r[_i];
MPRINT(CTWRET):   end;
MPRINT(CTWRET):   end;
```

```
MPRINT(CTWRET):   output _r7;
MPRINT(CTWRET):   ;
MPRINT(CTWRET):   end;
MPRINT(CTWRET):   ;
MPRINT(CTWRET):   run;

NOTE: There were 1 observations read from the data set WORK._R3.
NOTE: The data set WORK._R7 has 36 observations and 3 variables.
NOTE: DATA statement used:
      real time          0.02 seconds
      cpu time           0.02 seconds


MPRINT(CTWRET):   proc sort data=_r7;
MPRINT(CTWRET):   by date cticker;

NOTE: There were 36 observations read from the data set WORK._R7.
NOTE: The data set WORK._R7 has 36 observations and 3 variables.
NOTE: PROCEDURE SORT used:
      real time          0.01 seconds
      cpu time           0.01 seconds


MPRINT(CTWRET):   proc transpose data=_r7 out=_dtt(keep = _numeric_);
MPRINT(CTWRET):   var return;
MPRINT(CTWRET):   id cticker;
MPRINT(CTWRET):   by date;
MPRINT(CTWRET):   run;

NOTE: There were 36 observations read from the data set WORK._R7.
NOTE: The data set WORK._DTT has 36 observations and 2 variables.
NOTE: PROCEDURE TRANSPOSE used:
      real time          0.02 seconds
      cpu time           0.02 seconds


MPRINT(TERR_CTW):  ;
MPRINT(W2LTRAN):   proc transpose data=_dtt out=_dtr(keep = date ticker col1 rename=(col1=return)) name=ticker;
MPRINT(W2LTRAN):   by date;
MPRINT(W2LTRAN):   run;

NOTE: There were 36 observations read from the data set WORK._DTT.
NOTE: The data set WORK._DTR has 36 observations and 3 variables.
NOTE: PROCEDURE TRANSPOSE used:
      real time          0.02 seconds
      cpu time           0.01 seconds


MPRINT(W2LTRAN):   data _dtr;
MPRINT(W2LTRAN):   length ticker $8;
MPRINT(W2LTRAN):   set _dtr(where=(return ne .));
MPRINT(W2LTRAN):   run;

NOTE: There were 36 observations read from the data set WORK._DTR.
      WHERE return not = .;
NOTE: The data set WORK._DTR has 36 observations and 3 variables.
NOTE: DATA statement used:
      real time          0.01 seconds
```

```
      cpu time            0.02 seconds


MPRINT(DTT2DTR):  ;
MPRINT(TERR_CTW):  ;
MPRINT(TERR_CTW):   proc sql noprint;
MPRINT(TERR_CTW):   create table pf.mdl_terr as select ticker, std(return-_BENCH)*sqrt(12) as trackerr from _dtr a, pf.mdl_dtt b
where a.date=b.date group by ticker order by ticker;
NOTE: Table PF.MDL_TERR created, with 1 rows and 2 columns.

MPRINT(OPTO_PF):  ;
Optimization succeeded
MPRINT(DBEXEC):  ;
NOTE: PROCEDURE SQL used:
      real time           0.03 seconds
      cpu time            0.02 seconds


MPRINT(DBEXEC):   proc sql noerrorstop;
MPRINT(DBEXEC):   connect to sybase as pod( server=SYBCAUCHY database=pod user=wateradm pass=wateradm );
MPRINT(DBEXEC):   execute (pop2q_setstatus 'fpp/PRODUCTION/200203', 'FPP05003/2/10', 'done') by pod;
MPRINT(SETSTATUS):  ;
MPRINT(OPTO_PF):  ;
11789
NOTE: PROCEDURE SQL used:
      real time           0.14 seconds
      cpu time            0.08 seconds


NOTE: SAS Institute Inc., SAS Campus Drive, Cary, NC USA 27513-2414
NOTE: The SAS System used:
      real time           23.82 seconds
      cpu time            7.89 seconds
```

PROC NLP: Nonlinear Minimization

NOTE: Initial point was changed to be feasible for boundary and linear constraints.

Exhibit D

PROC NLP: Nonlinear Minimization

Optimization Start
Parameter Estimates

N	Parameter	Estimate	Gradient Objective Function	Lower Bound Constraint	Upper Bound Constraint	Active Bound Constraint
1	CPO	0.100000	23.561196	0.100000	0.100000	Equal BC
2	_BOND	0.209167	17.327263	0.112500	1.000000	
3	_CASH_	0.209167	16.425350	0	1.000000	
4	_EAFE	0.180000	2.759056	0	0.180000	Upper BC
5	_ML_HYM	0.092500	13.147478	0	0.100000	
6	_W5000	0.209167	-4.304211	0	1.000000	

Value of Objective Function = 5.1142576555

Linear Constraints

1	-8.327E-17	: ACT	1.0000	==	+	1.0000 * CPO	+	1.0000 * _BOND	+	1.0000 * _CASH_	+	1.0000 * _EAFE	+	1.0000 * _ML_HYM + 1.0000 * _W5000
2	0.30917	:	0	<=	+	1.0000 * CPO	+	1.0000 * _W5000						
3	0.63083	:	0.9400	>=	+	1.0000 * CPO	+	1.0000 * _W5000						
4	0.48917	:	0	<=	+	1.0000 * CPO	+	1.0000 * _EAFE	+	1.0000 * _W5000				
5	0.40333	:	0.8925	>=	+	1.0000 * CPO	+	1.0000 * _EAFE	+	1.0000 * _W5000				

PROC NLP: Nonlinear Minimization

Null Space Method of Quadratic Problem

Parameter Estimates	6
Lower Bounds	6
Upper Bounds	6
Linear Constraints	5

Optimization Start

Active Constraints	3	Objective Function	5.1142576555
Max Abs Gradient Element	17.266446025		

Iter	Restarts	Function Calls	Active Constraints	Objective Function	Objective Function Change	Max Abs Gradient Element	Step Size	Slope of Search Direction
1	0	2	4	1.02847	4.0858	1.2405	0.922	-8.221
2	0	3	5	1.02187	0.00660	1.0173	0.144	-0.0492
3	0	4	5	1.00583	0.0160	1.4148	0.688	-0.0356
4	0	5	4	0.87553	0.1303	1.0785	1.000	-0.261
5	0	6	3	0.85664	0.0189	0.0356	1.000	-0.0378
6	0	7	3	0.85609	0.000544	1.36E-15	1.000	-0.0011

Optimization Results

Iterations	6	Function Calls	8
Gradient Calls	7	Active Constraints	3
Objective Function	0.8560941258	Max Abs Gradient Element	9.064933E-16
Slope of Search Direction	-0.001088002		

ABSGCONV convergence criterion satisfied.

PROC NLP: Nonlinear Minimization

Optimization Results
Parameter Estimates

N	Parameter	Estimate	Approx Std Err	t Value	Approx Pr > \|t\|	Gradient Objective Function	Active Bound Constraint
1	CPO	0.100000	0	9999.990000	0.000063662	17.121883	Equal BC
2	_BOND	0.116470	0.761171	0.153015	0.903337	-1.77636E-15	
3	_CASH_	0.026496	0.803287	0.032985	0.979009	-8.88178E-16	
4	_EAFE	0.023428	0.288107	0.081318	0.948345	-2.22045E-16	
5	_ML_HYM	0	0	0	1.000000	0.046529	Lower BC
6	_W5000	0.733605	0.256966	2.854870	0.214492	0	

Value of Objective Function = 0.8560941258

Linear Constraints Evaluated at Solution

1	ACT	0	=	-1.0000	+	1.0000 * CPO	+	1.0000 * _BOND	+	1.0000 * _CASH_	+	1.0000 * _EAFE	+	1.0000 * _ML_HYM
	+	1.0000 * _W5000												
2		0.83361	=	0	+	1.0000 * CPO	+	1.0000 * _W5000						
3		0.10639	=	0.9400	-	1.0000 * CPO	-	1.0000 * _W5000						
4		0.85703	=	0	+	1.0000 * CPO	+	1.0000 * _EAFE	+	1.0000 * _W5000				
5		0.03547	=	0.8925	-	1.0000 * CPO	-	1.0000 * _EAFE	-	1.0000 * _W5000				

RHS
0.856094

PROC NLP: Nonlinear Minimization

Deleted Identical Linear Constraints

0.100000 >=

NOTE: Initial point was changed to be feasible for boundary and linear constraints.

PROC NLP: Nonlinear Minimization

Optimization Start
Parameter Estimates

N	Parameter	Estimate	Gradient Objective Function	Lower Bound Constraint	Upper Bound Constraint	Active Bound Constraint
1	CPO	0.100000	11.383390	0.100000	0.100000	Equal BC
2	FBNDX	0.078351	4.403253	0	1.000000	
3	FCNTX	0.550993	3.018354	0	1.000000	
4	FGRIX	0.270656	1.954756	0	1.000000	
5	FLPSX	0	0.756843	0	1.000000	Lower BC
6	FMAGX	0	-1.114877	0	1.000000	Lower BC
7	FOSFX	0	-1.336266	0	1.000000	Lower BC
8	FUSEX	0	-1.175001	0	1.000000	Lower BC
9	_GIC_	0	3.731108	0	1.000000	Lower BC

Value of Objective Function = 1.7219712407

Linear Constraints

1 4.4409E-16 : ACT 1.0000 == + 1.0000 * CPO + 1.0000 * FBNDX + 1.0000 * FCNTX + 1.0000 * FGRIX + 1.0000 * FLPSX + 1.0000 * FMAGX + 1.0000 * FOSFX + 1.0000 * FUSEX + 1.0000 * _GIC_

2 0.05278 : 0.0165 <= + 0.8603 * FBNDX + 0.00320 * FCNTX + 0.000300 * FGRIX + 0.0146 * FOSFX + 0.8000 * _GIC_

3 0.14722 : 0.2165 >= + 0.8603 * FBNDX + 0.00320 * FCNTX + 0.000300 * FGRIX + 0.0146 * FOSFX + 0.8000 * _GIC_

4 0.14507 : -0.0735 <= + 0.1397 * FBNDX + 0.0581 * FCNTX + 0.1057 * FGRIX + 0.1915 * FLPSX + 0.0561 * FMAGX + 0.0814 * FOSFX + 0.00830 * FUSEX + 0.2000 * _GIC_

5 0.05493 : 0.1265 >= + 0.1397 * FBNDX + 0.0581 * FCNTX + 0.1057 * FGRIX + 0.1915 * FLPSX + 0.0561 * FMAGX + 0.0814 * FOSFX + 0.00830 * FUSEX + 0.2000 * _GIC_

6 0.16000 : -0.0566 <= + 0.1800 * FCNTX + 0.0157 * FGRIX + 0.1304 * FLPSX + 0.0277 * FMAGX + 0.7889 * FOSFX + 0.0193 * FUSEX

7 1.7347E-17 : ACT 0.1034 >= + 0.1800 * FCNTX + 0.0157 * FGRIX + 0.1304 * FLPSX + 0.0277 * FMAGX + 0.7889 * FOSFX + 0.0193 * FUSEX

8 0.10000 : 0.1000 >=

9 0.11215 : -0.1000 <= + 0.00540 * FCNTX + 0.0339 * FGRIX + 0.00480 * FLPSX + 0.0294 * FOSFX + 0.00890 * FUSEX

10 0.08785 : 0.1000 >= + 0.00540 * FCNTX + 0.0339 * FGRIX + 0.00480 * FLPSX + 0.0294 * FOSFX + 0.00890 * FUSEX

11 -8.327E-17 : ACT 0.8470 <= + 1.0000 * CPO + 0.9333 * FCNTX + 0.8601 * FGRIX + 0.8037 * FLPSX + 0.9439 * FMAGX + 0.8746 * FOSFX + 0.9828 * FUSEX

12 0.02000 : 0.8670 >= + 1.0000 * CPO + 0.9333 * FCNTX + 0.8601 * FGRIX + 0.8037 * FLPSX + 0.9439 * FMAGX + 0.8746 * FOSFX + 0.9828 * FUSEX

PROC NLP: Nonlinear Minimization

Null Space Method of Quadratic Problem

Parameter Estimates	9
Lower Bounds	9
Upper Bounds	9
Linear Constraints	12

Optimization Start

Active Constraints (+)	8	Objective Function	1.7219712407
Max Abs Gradient Element	1.455060137		

Iter	Restarts	Function Calls	Active Constraints	Objective Function	Objective Function Change	Max Abs Gradient Element	Step Size	Slope of Search Direction
1	0	2	7'	1.39796	0.3240	2.5134	1.000	-0.648
2	0	3	6'	0.46385	0.9341	1.0729	1.000	-1.868
3	0	4	5'	0.38259	0.0813	0.6061	1.000	-0.163
4	0	5	6	0.35422	0.0284	0.4944	0.141	-0.216
5	0	6	5'	0.29360	0.0606	0.6479	1.000	-0.121
6	0	7	5'	0.28136	0.0122	0.3710	0.464	-0.0343
7	0	8	6	0.25006	0.0313	0.0438	0.395	-0.0987
8	0	9	5'	0.24989	0.000169	0.0523	1.000	-0.0003
9	0	10	5	0.24977	0.000118	2.29E-16	1.000	-0.0002

Optimization Results

Iterations	9	Function Calls	11
Gradient Calls	10	Active Constraints	5
Objective Function	0.2497684956	Max Abs Gradient Element	2.775558E-16
Slope of Search Direction	-0.000235093		

ABSGCONV convergence criterion satisfied.

PROC NLP: Nonlinear Minimization

Optimization Results
Parameter Estimates

N	Parameter	Estimate	Approx Std Err	t Value	Approx Pr > \|t\|	Gradient Objective Function	Active Bound Constraint
1	CPO	0.100000	0	9999.990000	0.000063662	9.134575	Equal BC
2	FBNDX	0	0	0	1.000000	0.601800	Lower BC
3	FCNTX	0.040974	0.452121	0.090627	0.942462	-0.015954	
4	FGRIX	0	0	0	1.000000	0.604141	Lower BC
5	FLPSX	0.101853	0.365187	0.278905	0.826844	0.068897	
6	FMAGX	0.334913	0.608459	0.550428	0.679671	0.244587	
7	FOSFX	0.086911	0.102258	0.849918	0.551535	-1.057572	
8	FUSEX	0.255534	0.584214	0.437398	0.737506	0.258957	
9	_GIC_	0.079815	0.268879	0.296845	0.816297	0.291974	

Value of Objective Function = 0.2497684956

Linear Constraints Evaluated at Solution

1 ACT 2.7756E-16 = -1.0000 + 1.0000 * CPO + 1.0000 * FBNDX + 1.0000 * FCNTX + 1.0000 * FGRIX + 1.0000 * FLPSX + 1.0000 * FMAGX + 1.0000 * FOSFX + 1.0000 * FUSEX + 1.0000 * _GIC_

2 0.04878 = -0.0165 + 0.8603 * FBNDX + 0.00320 * FCNTX + 0.000300 * FGRIX + 0.0146 * FOSFX + 0.8000 * _GIC_

3 0.15122 = 0.2165 - 0.8603 * FBNDX - 0.00320 * FCNTX - 0.000300 * FGRIX - 0.0146 * FOSFX - 0.8000 * _GIC_

4 0.13933 = 0.0735 + 0.1397 * FBNDX + 0.0581 * FCNTX + 0.1057 * FGRIX + 0.1915 * FLPSX + 0.0561 * FMAGX + 0.0814 * FOSFX + 0.00830 * FUSEX + 0.2000 * _GIC_

5 0.06066 = 0.1265 - 0.1397 * FBNDX - 0.0581 * FCNTX - 0.1057 * FGRIX - 0.1915 * FLPSX - 0.0561 * FMAGX - 0.0814 * FOSFX - 0.00830 * FUSEX - 0.2000 * _GIC_

6 0.16000 = 0.0566 + 0.1800 * FCNTX + 0.0157 * FGRIX + 0.1304 * FLPSX + 0.0277 * FMAGX + 0.7889 * FOSFX + 0.0193 * FUSEX

7 ACT 2.6888E-17 = 0.1034 - 0.1800 * FCNTX - 0.0157 * FGRIX - 0.1304 * FLPSX - 0.0277 * FMAGX - 0.7889 * FOSFX - 0.0193 * FUSEX

8 0.10000 = 0.1000

9 0.10554 = 0.1000 + 0.00540 * FCNTX + 0.0339 * FGRIX + 0.00480 * FLPSX + 0.0294 * FOSFX + 0.00890 * FUSEX

10 0.09446 = 0.1000 - 0.00540 * FCNTX - 0.0339 * FGRIX - 0.00480 * FLPSX - 0.0294 * FOSFX - 0.00890 * FUSEX

11 0.01634 = -0.8470 + 1.0000 * CPO + 0.9333 * FCNTX + 0.8601 * FGRIX + 0.8037 * FLPSX + 0.9439 * FMAGX + 0.8746 * FOSFX + 0.9828 * FUSEX

12 0.00366 = 0.8670 - 1.0000 * CPO - 0.9333 * FCNTX - 0.8601 * FGRIX - 0.8037 * FLPSX - 0.9439 * FMAGX - 0.8746 * FOSFX - 0.9828 * FUSEX

RHS
0.249768

0.040974

1

Iteration 1 minobjfn = 99999 last objfnval =

PROC NLP: Nonlinear Minimization

Deleted Identical Linear Constraints

0.100000 >=

NOTE: Initial point was changed to be feasible for boundary and linear constraints.

PROC NLP: Nonlinear Minimization

N	Parameter	Optimization Start Parameter Estimates Estimate	Gradient Objective Function	Lower Bound Constraint	Upper Bound Constraint	Active Bound Constraint
1	CPO	0.100000	8.575135	0.100000	0.100000	Equal BC
2	FBNDX	0.115698	1.290136	0	1.000000	
3	FCNTX	0.168840	0.569871	0	1.000000	
4	FGRIX	0	0.965063	0	1.000000	Lower BC
5	FLPSX	0	-0.390853	0	1.000000	Lower BC
6	FMAGX	0.396430	0.379284	0	1.000000	
7	FOSFX	-2.77556E-17	-1.562442	0	1.000000	Lower BC
8	FUSEX	0.219032	0.218568	0	1.000000	
9	_GIC_	-5.55112E-17	0.669596	0	1.000000	Lower BC

Value of Objective Function = 0.4969787227

Linear Constraints

1 1.1102E-16 : ACT 1.0000 == + 1.0000 * CPO + 1.0000 * FBNDX + 1.0000 * FCNTX + 1.0000 * FGRIX + 1.0000 * FLPSX + 1.0000 * FMAGX + 1.0000 * FOSFX + 1.0000 * FUSEX + 1.0000 * _GIC_

2 0.08361 : 0.0165 <= + 0.8603 * FBNDX + 0.00320 * FCNTX + 0.000300 * FGRIX + 0.0146 * FOSFX + 0.8000 * _GIC_

3 0.11639 : 0.2165 >= + 0.8603 * FBNDX + 0.00320 * FCNTX + 0.000300 * FGRIX + 0.0146 * FOSFX + 0.8000 * _GIC_

4 0.12353 : -0.0735 <= + 0.1397 * FBNDX + 0.0581 * FCNTX + 0.1057 * FGRIX + 0.1915 * FLPSX + 0.0561 * FMAGX + 0.0814 * FOSFX + 0.00830 * FUSEX + 0.2000 * _GIC_

5 0.07647 : 0.1265 >= + 0.1397 * FBNDX + 0.0581 * FCNTX + 0.1057 * FGRIX + 0.1915 * FLPSX + 0.0561 * FMAGX + 0.0814 * FOSFX + 0.00830 * FUSEX + 0.2000 * _GIC_

6 0.10217 : -0.0566 <= + 0.1800 * FCNTX + 0.0157 * FGRIX + 0.1304 * FLPSX + 0.0277 * FMAGX + 0.7889 * FOSFX + 0.0193 * FUSEX

7 0.05783 : 0.1034 >= + 0.1800 * FCNTX + 0.0157 * FGRIX + 0.1304 * FLPSX + 0.0277 * FMAGX + 0.7889 * FOSFX + 0.0193 * FUSEX

8 0.10000 : 0.1000 >=

9 0.10286 : -0.1000 <= + 0.00540 * FCNTX + 0.0339 * FGRIX + 0.00480 * FLPSX + 0.0294 * FOSFX + 0.00890 * FUSEX

10 0.09714 : 0.1000 >= + 0.00540 * FCNTX + 0.0339 * FGRIX + 0.00480 * FLPSX + 0.0294 * FOSFX + 0.00890 * FUSEX

11 -2.776E-17 : ACT 0.8470 <= + 1.0000 * CPO + 0.9333 * FCNTX + 0.8601 * FGRIX + 0.8037 * FLPSX + 0.9439 * FMAGX + 0.8746 * FOSFX + 0.9828 * FUSEX

12 0.02000 : 0.8670 >= + 1.0000 * CPO + 0.9333 * FCNTX + 0.8601 * FGRIX + 0.8037 * FLPSX + 0.9439 * FMAGX + 0.8746 * FOSFX + 0.9828 * FUSEX

PROC NLP: Nonlinear Minimization

Null Space Method of Quadratic Problem

Parameter Estimates	9
Lower Bounds	9
Upper Bounds	9
Linear Constraints	12

Optimization Start

Active Constraints (+)	6	Objective Function	0.4969787227
Max Abs Gradient Element	1.7842596228		

Iter	Restarts	Function Calls	Active Constraints	Objective Function	Objective Function Change	Max Abs Gradient Element	Step Size	Slope of Search Direction
1	0	2	7	0.34831	0.1487	0.3396	0.390	-0.474
2	0	3	6'	0.33689	0.0114	0.7729	1.000	-0.0228
3	0	4	5'	0.29360	0.0433	0.6479	1.000	-0.0866
4	0	5	5'	0.28136	0.0122	0.3710	0.464	-0.0343
5	0	6	6	0.25006	0.0313	0.0438	0.395	-0.0987
6	0	7	5'	0.24989	0.000169	0.0523	1.000	-0.0003
7	0	8	5	0.24977	0.000118	1.11E-16	1.000	-0.0002

Optimization Results

Iterations	7	Function Calls	9
Gradient Calls	8	Active Constraints	5
Objective Function	0.2497684956	Max Abs Gradient Element	1.110223E-16
Slope of Search Direction	-0.000235093		

ABSGCONV convergence criterion satisfied.

PROC NLP: Nonlinear Minimization

Optimization Results
Parameter Estimates

N	Parameter	Estimate	Approx Std Err	t Value	Approx Pr > \|t\|	Gradient Objective Function	Active Bound Constraint
1	CPO	0.100000	0	9999.990000	0.000063662	9.134575	Equal BC
2	FBNDX	0	0	0	1.000000	0.601800	Lower BC
3	FCNTX	0.040974	0.452121	0.090627	0.942462	-0.015954	
4	FGRIX	0	0	0	1.000000	0.604141	Lower BC
5	FLPSX	0.101853	0.365187	0.278905	0.826844	0.068897	
6	FMAGX	0.334913	0.608459	0.550428	0.679671	0.244587	
7	FOSFX	0.086911	0.102258	0.849918	0.551535	-1.057572	
8	FUSEX	0.255534	0.584214	0.437398	0.737506	0.258957	
9	_GIC_	0.079815	0.268879	0.296845	0.816297	0.291974	

Value of Objective Function = 0.2497684956

Linear Constraints Evaluated at Solution

1	ACT	0	=	-1.0000	+	1.0000 * CPO	+	1.0000 * FBNDX	+	1.0000 * FCNTX	+	1.0000 * FGRIX	+ 1.0000 * FLPSX + 1.0000 * FMAGX + 1.0000 * FOSFX + 1.0000 * FUSEX + 1.0000 * _GIC_
2		0.04878	=	-0.0165	+	0.8603 * FBNDX	+	0.00320 * FCNTX	+	0.000300 * FGRIX	+	0.0146 * FOSFX	+ 0.8000 * _GIC_
3		0.15122	=	0.2165	-	0.8603 * FBNDX	-	0.00320 * FCNTX	-	0.000300 * FGRIX	-	0.0146 * FOSFX	- 0.8000 * _GIC_
4		0.13933	=	0.0735	+	0.1397 * FBNDX	+	0.0581 * FCNTX	+	0.1057 * FGRIX	+	0.1915 * FLPSX	+ 0.0561 * FMAGX + 0.0814 * FOSFX + 0.00830 * FUSEX + 0.2000 * _GIC_
5		0.06066	=	0.1265	-	0.1397 * FBNDX	-	0.0581 * FCNTX	-	0.1057 * FGRIX	-	0.1915 * FLPSX	- 0.0561 * FMAGX - 0.0814 * FOSFX - 0.00830 * FUSEX - 0.2000 * _GIC_
6		0.16000	=	0.0566	+	0.1800 * FCNTX	+	0.0157 * FGRIX	+	0.1304 * FLPSX	+	0.0277 * FMAGX	+ 0.7889 * FOSFX + 0.0193 * FUSEX
7	ACT	1.301E-17	=	0.1034	-	0.1800 * FCNTX	-	0.0157 * FGRIX	-	0.1304 * FLPSX	-	0.0277 * FMAGX	- 0.7889 * FOSFX - 0.0193 * FUSEX
8		0.10000	=	0.1000									
9		0.10554	=	0.1000	+	0.00540 * FCNTX	+	0.0339 * FGRIX	+	0.00480 * FLPSX	+	0.0294 * FOSFX	+ 0.00890 * FUSEX
10		0.09446	=	0.1000	-	0.00540 * FCNTX	-	0.0339 * FGRIX	-	0.00480 * FLPSX	-	0.0294 * FOSFX	- 0.00890 * FUSEX
11		0.01634	=	-0.8470	+	1.0000 * CPO	+	0.9333 * FCNTX	+	0.8601 * FGRIX	+	0.8037 * FLPSX	+ 0.9439 * FMAGX + 0.8746 * FOSFX + 0.9828 * FUSEX
12		0.00366	=	0.8670	-	1.0000 * CPO	-	0.9333 * FCNTX	-	0.8601 * FGRIX	-	0.8037 * FLPSX	- 0.9439 * FMAGX - 0.8746 * FOSFX - 0.9828 * FUSEX

RHS
0.249768

```
--------
0.249768
```

_force

Obs	TICKER	WGT	LOWER	UPPER
1	FBNDX	0.00000	0.0	1.0
2	FGRIX	0.00000	0.0	1.0
3	FCNTX	0.04097	0.0	1.0
4	_GIC_	0.07982	0.0	1.0
5	FOSFX	0.08691	0.0	1.0
6	CPO	0.10000	0.1	0.1
7	FLPSX	0.10185	0.0	1.0
8	FUSEX	0.25553	0.0	1.0
9	FMAGX	0.33491	0.0	1.0

Obs	TICKER	WGT	LOWER	UPPER
1	FBNDX	0.00000	0.00	0.0
2	FGRIX	0.00000	0.00	0.0
3	FCNTX	0.04097	0.05	1.0
4	_GIC_	0.07982	0.05	1.0
5	FOSFX	0.08691	0.05	1.0
6	CPO	0.10000	0.10	0.1
7	FLPSX	0.10185	0.05	1.0
8	FUSEX	0.25553	0.05	1.0
9	FMAGX	0.33491	0.05	1.0

PROC NLP: Nonlinear Minimization

Deleted Identical Linear Constraints

0.100000 >=

NOTE: Initial point was changed to be feasible for boundary and linear constraints.

PROC NLP: Nonlinear Minimization

N	Parameter	Optimization Start Parameter Estimates Estimate	Gradient Objective Function	Lower Bound Constraint	Upper Bound Constraint	Active Bound Constraint
1	CPO	0.100000	8.413943	0.100000	0.100000	Equal BC
2	FBNDX	0	-0.158807	0	0	Equal BC
3	FCNTX	0.050000	0.075162	0.050000	1.000000	Lower BC
4	FGRIX	0	0.398700	0	0	Equal BC
5	FLPSX	0.050000	-0.593725	0.050000	1.000000	Lower BC
6	FMAGX	0.585479	0.756978	0.050000	1.000000	
7	FOSFX	0.089653	-0.757605	0.050000	1.000000	
8	FUSEX	0.050000	0.005090	0.050000	1.000000	Lower BC
9	_GIC_	0.074868	-0.485687	0.050000	1.000000	

Value of Objective Function = 0.3599771115

Linear Constraints

1 2.498E-16 : ACT 1.0000 == + 1.0000 * CPO + 1.0000 * FBNDX + 1.0000 * FCNTX + 1.0000 * FGRIX + 1.0000 * FLPSX + 1.0000 * FMAGX + 1.0000 * FOSFX + 1.0000 * FUSEX + 1.0000 * _GIC_

2 0.04489 : 0.0165 <= + 0.8603 * FBNDX + 0.00320 * FCNTX + 0.000300 * FGRIX + 0.0146 * FOSFX + 0.8000 * _GIC_

3 0.15511 : 0.2165 >= + 0.8603 * FBNDX + 0.00320 * FCNTX + 0.000300 * FGRIX + 0.0146 * FOSFX + 0.8000 * _GIC_

4 0.14151 : -0.0735 <= + 0.1397 * FBNDX + 0.0581 * FCNTX + 0.1057 * FGRIX + 0.1915 * FLPSX + 0.0561 * FMAGX + 0.0814 * FOSFX + 0.00830 * FUSEX + 0.2000 * _GIC_

5 0.05848 : 0.1265 >= + 0.1397 * FBNDX + 0.0581 * FCNTX + 0.1057 * FGRIX + 0.1915 * FLPSX + 0.0561 * FMAGX + 0.0814 * FOSFX + 0.00830 * FUSEX + 0.2000 * _GIC_

6 0.16000 : -0.0566 <= + 0.1800 * FCNTX + 0.0157 * FGRIX + 0.1304 * FLPSX + 0.0277 * FMAGX + 0.7889 * FOSFX + 0.0193 * FUSEX

7 7.5894E-18 : ACT 0.1034 >= + 0.1800 * FCNTX + 0.0157 * FGRIX + 0.1304 * FLPSX + 0.0277 * FMAGX + 0.7889 * FOSFX + 0.0193 * FUSEX

8 0.10000 : 0.1000 >=

9 0.10359 : -0.1000 <= + 0.00540 * FCNTX + 0.0339 * FGRIX + 0.00480 * FLPSX + 0.0294 * FOSFX + 0.00890 * FUSEX

10 0.09641 : 0.1000 >= + 0.00540 * FCNTX + 0.0339 * FGRIX + 0.00480 * FLPSX + 0.0294 * FOSFX + 0.00890 * FUSEX

11 0.02000 : 0.8470 <= + 1.0000 * CPO + 0.9333 * FCNTX + 0.8601 * FGRIX + 0.8037 * FLPSX + 0.9439 * FMAGX + 0.8746 * FOSFX + 0.9828 * FUSEX

12 2.7756E-17 : ACT 0.8670 >= + 1.0000 * CPO + 0.9333 * FCNTX + 0.8601 * FGRIX + 0.8037 * FLPSX + 0.9439 * FMAGX + 0.8746 * FOSFX + 0.9828 * FUSEX

PROC NLP: Nonlinear Minimization

Null Space Method of Quadratic Problem

Parameter Estimates	9
Lower Bounds	9
Upper Bounds	9
Linear Constraints	12

Optimization Start

Active Constraints (+)	8	Objective Function	0.3599771115
Max Abs Gradient Element	0.9008402961		

Iter	Restarts	Function Calls	Active Constraints	Objective Function	Objective Function Change	Max Abs Gradient Element	Step Size	Slope of Search Direction
1	0	2	8'	0.33909	0.0209	0.6148	0.477	-0.0575
2	0	3	7'	0.31213	0.0270	0.4581	1.000	-0.0539
3	0	4	6'	0.25374	0.0584	0.3220	1.000	-0.117
4	0	5	6	0.24997	0.00377	8.33E-17	1.000	-0.0075

Optimization Results

Iterations	4	Function Calls	6
Gradient Calls	5	Active Constraints	6
Objective Function	0.2499677559	Max Abs Gradient Element	2.220446E-16
Slope of Search Direction	-0.007540479		

ABSGCONV convergence criterion satisfied.

PROC NLP: Nonlinear Minimization

Optimization Results
Parameter Estimates

N	Parameter	Estimate	Approx Std Err	t Value	Approx Pr > \|t\|	Gradient Objective Function	Active Bound Constraint
1	CPO	0.100000	0	9999.990000	0.000063662	9.146782	Equal BC
2	FBNDX	0	0	0	1.000000	0.596713	Equal BC
3	FCNTX	0.050000	0	9999.990000	0.000063662	0.021798	Lower BC
4	FGRIX	0	0	0	1.000000	0.605292	Equal BC
5	FLPSX	0.100616	0.359891	0.279572	0.826450	0.061883	
6	FMAGX	0.330541	0.567678	0.582268	0.664324	0.236304	
7	FOSFX	0.085201	0.055857	1.525329	0.369430	-1.056450	
8	FUSEX	0.255882	0.583954	0.438189	0.737083	0.250570	
9	_GIC_	0.077761	0.248400	0.313046	0.806861	0.283349	

Value of Objective Function = 0.2499677559

Linear Constraints Evaluated at Solution

1 ACT 4.4409E-16 = -1.0000 + 1.0000 * CPO + 1.0000 * FBNDX + 1.0000 * FCNTX + 1.0000 * FGRIX + 1.0000 * FLPSX + 1.0000 * FMAGX + 1.0000 * FOSFX + 1.0000 * FUSEX + 1.0000 * _GIC_

2 0.04714 = -0.0165 + 0.8603 * FBNDX + 0.00320 * FCNTX + 0.000300 * FGRIX + 0.0146 * FOSFX + 0.8000 * _GIC_

3 0.15286 = 0.2165 - 0.8603 * FBNDX - 0.00320 * FCNTX - 0.000300 * FGRIX - 0.0146 * FOSFX - 0.8000 * _GIC_

4 0.13883 = 0.0735 + 0.1397 * FBNDX + 0.0581 * FCNTX + 0.1057 * FGRIX + 0.1915 * FLPSX + 0.0561 * FMAGX + 0.0814 * FOSFX + 0.00830 * FUSEX + 0.2000 * _GIC_

5 0.06117 = 0.1265 - 0.1397 * FBNDX - 0.0581 * FCNTX - 0.1057 * FGRIX - 0.1915 * FLPSX - 0.0561 * FMAGX - 0.0814 * FOSFX - 0.00830 * FUSEX - 0.2000 * _GIC_

6 0.16000 = 0.0566 + 0.1800 * FCNTX + 0.0157 * FGRIX + 0.1304 * FLPSX + 0.0277 * FMAGX + 0.7889 * FOSFX + 0.0193 * FUSEX

7 ACT -5.204E-18 = 0.1034 - 0.1800 * FCNTX - 0.0157 * FGRIX - 0.1304 * FLPSX - 0.0277 * FMAGX - 0.7889 * FOSFX - 0.0193 * FUSEX

8 0.10000 = 0.1000

9 0.10554 = 0.1000 + 0.00540 * FCNTX + 0.0339 * FGRIX + 0.00480 * FLPSX + 0.0294 * FOSFX + 0.00890 * FUSEX

10 0.09446 = 0.1000 - 0.00540 * FCNTX - 0.0339 * FGRIX - 0.00480 * FLPSX - 0.0294 * FOSFX - 0.00890 * FUSEX

11 0.01849 = -0.8470 + 1.0000 * CPO + 0.9333 * FCNTX + 0.8601 * FGRIX + 0.8037 * FLPSX + 0.9439 * FMAGX + 0.8746 * FOSFX + 0.9828 * FUSEX

12 0.00151 = 0.8670 - 1.0000 * CPO - 0.9333 * FCNTX - 0.8601 * FGRIX - 0.8037 * FLPSX - 0.9439 * FMAGX - 0.8746 * FOSFX - 0.9828 * FUSEX

RHS
0.249968

2

Iteration 2 minobjfn = 0.2499677559 last objfnval = 0.2499677559

PROC NLP: Nonlinear Minimization

Deleted Identical Linear Constraints

0.100000 >=

NOTE: Initial point was changed to be feasible for boundary and linear constraints.

PROC NLP: Nonlinear Minimization

Optimization Start
Parameter Estimates

N	Parameter	Estimate	Gradient Objective Function	Lower Bound Constraint	Upper Bound Constraint	Active Bound Constraint
1	CPO	0.100000	11.650796	0.100000	0.100000	Equal BC
2	FBNDX	0.018669	4.171430	0	1.000000	
3	FCNTX	0.050000	0.999077	0.050000	1.000000	Lower BC
4	FGRIX	0.831331	3.806933	0	1.000000	
5	FLPSX	0	0.451713	0	1.000000	Lower BC
6	FMAGX	0	-0.392199	0	1.000000	Lower BC
7	FOSFX	0	-2.786950	0	1.000000	Lower BC
8	FUSEX	0	0.237452	0	1.000000	Lower BC
9	_GIC_	2.775558E-17	3.696729	0	1.000000	Lower BC

Value of Objective Function = 2.0749583504

Linear Constraints

1 3.6082E-16 : ACT 1.0000 == + 1.0000 * CPO + 1.0000 * FBNDX + 1.0000 * FCNTX + 1.0000 * FGRIX + 1.0000 * FLPSX + 1.0000 * FMAGX + 1.0000 * FOSFX + 1.0000 * FUSEX + 1.0000 * _GIC_

2 4.12E-19 : ACT 0.0165 <= + 0.8603 * FBNDX + 0.00320 * FCNTX + 0.000300 * FGRIX + 0.0146 * FOSFX + 0.8000 * _GIC_

3 0.20000 : 0.2165 >= + 0.8603 * FBNDX + 0.00320 * FCNTX + 0.000300 * FGRIX + 0.0146 * FOSFX + 0.8000 * _GIC_

4 0.16688 : -0.0735 <= + 0.1397 * FBNDX + 0.0581 * FCNTX + 0.1057 * FGRIX + 0.1915 * FLPSX + 0.0561 * FMAGX + 0.0814 * FOSFX + 0.00830 * FUSEX + 0.2000 * _GIC_

5 0.03311 : 0.1265 >= + 0.1397 * FBNDX + 0.0581 * FCNTX + 0.1057 * FGRIX + 0.1915 * FLPSX + 0.0561 * FMAGX + 0.0814 * FOSFX + 0.00830 * FUSEX + 0.2000 * _GIC_

6 0.07862 : -0.0566 <= + 0.1800 * FCNTX + 0.0157 * FGRIX + 0.1304 * FLPSX + 0.0277 * FMAGX + 0.7889 * FOSFX + 0.0193 * FUSEX

7 0.08138 : 0.1034 >= + 0.1800 * FCNTX + 0.0157 * FGRIX + 0.1304 * FLPSX + 0.0277 * FMAGX + 0.7889 * FOSFX + 0.0193 * FUSEX

8 0.10000 : 0.1000 >=

9 0.12845 : -0.1000 <= + 0.00540 * FCNTX + 0.0339 * FGRIX + 0.00480 * FLPSX + 0.0294 * FOSFX + 0.00890 * FUSEX

10 0.07155 : 0.1000 >= + 0.00540 * FCNTX + 0.0339 * FGRIX + 0.00480 * FLPSX + 0.0294 * FOSFX + 0.00890 * FUSEX

11 0.01466 : 0.8470 <= + 1.0000 * CPO + 0.9333 * FCNTX + 0.8601 * FGRIX + 0.8037 * FLPSX + 0.9439 * FMAGX + 0.8746 * FOSFX + 0.9828 * FUSEX

12 0.00534 : 0.8670 >= + 1.0000 * CPO + 0.9333 * FCNTX + 0.8601 * FGRIX + 0.8037 * FLPSX + 0.9439 * FMAGX + 0.8746 * FOSFX + 0.9828 * FUSEX

PROC NLP: Nonlinear Minimization

Null Space Method of Quadratic Problem

Parameter Estimates	9
Lower Bounds	9
Upper Bounds	9
Linear Constraints	12

Optimization Start

Active Constraints (+)	8	Objective Function	2.0749583504
Max Abs Gradient Element	4.7054984602		

Iter	Restarts	Function Calls	Active Constraints	Objective Function	Objective Function Change	Max Abs Gradient Element	Step Size	Slope of Search Direction
1	0	2	8'	1.47446	0.6005	2.4891	0.270	-2.567
2	0	3	8'	1.37975	0.0947	2.2318	0.0515	-1.888
3	0	4	7'	0.47702	0.9027	1.2026	1.000	-1.805
4	0	5	6'	0.37740	0.0996	0.5946	1.000	-0.199
5	0	6	7	0.32677	0.0506	0.4014	0.267	-0.218
6	0	7	6'	0.28177	0.0450	0.3733	1.000	-0.0900
7	0	8	7	0.25015	0.0316	0.0435	0.396	-0.0995
8	0	9	6'	0.24999	0.000155	0.0263	1.000	-0.0003
9	0	10	6	0.24997	0.000025	2.64E-16	1.000	-502E-7

Optimization Results

Iterations	9	Function Calls	11
Gradient Calls	10	Active Constraints	6
Objective Function	0.2499677559	Max Abs Gradient Element	2.220446E-16
Slope of Search Direction	-0.000050238		

ABSGCONV convergence criterion satisfied.

PROC NLP: Nonlinear Minimization

Optimization Results
Parameter Estimates

N	Parameter	Estimate	Approx Std Err	t Value	Approx Pr > \|t\|	Gradient Objective Function	Active Bound Constraint
1	CPO	0.100000	0	9999.990000	0.000063662	9.146782	Equal BC
2	FBNDX	0	0	0	1.000000	0.596713	Lower BC
3	FCNTX	0.050000	0	9999.990000	0.000063662	0.021798	Lower BC
4	FGRIX	0	0	0	1.000000	0.605292	Lower BC
5	FLPSX	0.100616	0.359891	0.279572	0.826450	0.061883	
6	FMAGX	0.330541	0.567678	0.582268	0.664324	0.236304	
7	FOSFX	0.085201	0.055857	1.525329	0.369430	-1.056450	
8	FUSEX	0.255882	0.583954	0.438189	0.737083	0.250570	
9	_GIC_	0.077761	0.248400	0.313046	0.806861	0.283349	

Value of Objective Function = 0.2499677559

Linear Constraints Evaluated at Solution

1 ACT 1.6653E-16 = -1.0000 + 1.0000 * CPO + 1.0000 * FBNDX + 1.0000 * FCNTX + 1.0000 * FGRIX + 1.0000 * FLPSX + 1.0000 * FMAGX + 1.0000 * FOSFX + 1.0000 * FUSEX + 1.0000 * _GIC_

2 0.04714 = -0.0165 + 0.8603 * FBNDX + 0.00320 * FCNTX + 0.000300 * FGRIX + 0.0146 * FOSFX + 0.8000 * _GIC_

3 0.15286 = 0.2165 - 0.8603 * FBNDX - 0.00320 * FCNTX - 0.000300 * FGRIX - 0.0146 * FOSFX - 0.8000 * _GIC_

4 0.13883 = 0.0735 + 0.1397 * FBNDX + 0.0581 * FCNTX + 0.1057 * FGRIX + 0.1915 * FLPSX + 0.0561 * FMAGX + 0.0814 * FOSFX + 0.00830 * FUSEX + 0.2000 * _GIC_

5 0.06117 = 0.1265 - 0.1397 * FBNDX - 0.0581 * FCNTX - 0.1057 * FGRIX - 0.1915 * FLPSX - 0.0561 * FMAGX - 0.0814 * FOSFX - 0.00830 * FUSEX - 0.2000 * _GIC_

6 0.16000 = 0.0566 + 0.1800 * FCNTX + 0.0157 * FGRIX + 0.1304 * FLPSX + 0.0277 * FMAGX + 0.7889 * FOSFX + 0.0193 * FUSEX

7 ACT 2.2551E-17 = 0.1034 - 0.1800 * FCNTX - 0.0157 * FGRIX - 0.1304 * FLPSX - 0.0277 * FMAGX - 0.7889 * FOSFX - 0.0193 * FUSEX

8 0.10000 = 0.1000

9 0.10554 = 0.1000 + 0.00540 * FCNTX + 0.0339 * FGRIX + 0.00480 * FLPSX + 0.0294 * FOSFX + 0.00890 * FUSEX

10 0.09446 = 0.1000 - 0.00540 * FCNTX - 0.0339 * FGRIX - 0.00480 * FLPSX - 0.0294 * FOSFX - 0.00890 * FUSEX

11 0.01849 = -0.8470 + 1.0000 * CPO + 0.9333 * FCNTX + 0.8601 * FGRIX + 0.8037 * FLPSX + 0.9439 * FMAGX + 0.8746 * FOSFX + 0.9828 * FUSEX

12 0.00151 = 0.8670 - 1.0000 * CPO - 0.9333 * FCNTX - 0.8601 * FGRIX - 0.8037 * FLPSX - 0.9439 * FMAGX - 0.8746 * FOSFX - 0.9828 * FUSEX

Iteration 2 minobjfn = 0.2499677559 last objfnval = 0.2499677559

RHS
0.249968

Iteration 2 minobjfn = 0.2499677559 last objfnval = 0.2499677559

1

Iteration 3 minobjfn = 0.2499677559 last objfnval = 0.2499677559

PROC NLP: Nonlinear Minimization

Deleted Identical Linear Constraints

0.100000 >=

NOTE: Initial point was changed to be feasible for boundary and linear constraints.

PROC NLP: Nonlinear Minimization

Optimization Start
Parameter Estimates

N	Parameter	Estimate	Gradient Objective Function	Lower Bound Constraint	Upper Bound Constraint	Active Bound Constraint
1	CPO	0.100000	11.681492	0.100000	0.100000	Equal BC
2	FBNDX	0.018837	4.249768	0	1.000000	
3	FCNTX	0	0.808016	0	0	Equal BC
4	FGRIX	0.881163	3.989171	0	1.000000	
5	FLPSX	0	0.455036	0	1.000000	Lower BC
6	FMAGX	6.938894E-18	-0.342507	0	1.000000	Lower BC
7	FOSFX	0	-2.933409	0	1.000000	Lower BC
8	FUSEX	-2.77556E-17	0.357369	0	1.000000	Lower BC
9	_GIC_	0	3.783097	0	1.000000	Lower BC

Value of Objective Function = 2.2247362956

Linear Constraints

1	9.0206E-17	: ACT	1.0000	==	+ 1.0000 * CPO	+ 1.0000 * FBNDX	+ 1.0000 * FCNTX	+ 1.0000 * FGRIX	+ 1.0000 * FLPSX	+ 1.0000 * FMAGX	+ 1.0000 * FOSFX	+ 1.0000 * FUSEX + 1.0000 * _GIC_
2	-5.963E-19	: ACT	0.0165	<=	+ 0.8603 * FBNDX	+ 0.00320 * FCNTX	+ 0.000300 * FGRIX	+ 0.0146 * FOSFX	+ 0.8000 * _GIC_			
3		:	0.2165	>=	+ 0.8603 * FBNDX	+ 0.00320 * FCNTX	+ 0.000300 * FGRIX	+ 0.0146 * FOSFX	+ 0.8000 * _GIC_			
4		:	-0.0735	<=	+ 0.1397 * FBNDX	+ 0.0581 * FCNTX	+ 0.1057 * FGRIX	+ 0.1915 * FLPSX	+ 0.0561 * FMAGX	+ 0.0814 * FOSFX	+ 0.00830 * FUSEX	+ 0.2000 * _GIC_
5		:	0.1265	>=	+ 0.1397 * FBNDX	+ 0.0581 * FCNTX	+ 0.1057 * FGRIX	+ 0.1915 * FLPSX	+ 0.0561 * FMAGX	+ 0.0814 * FOSFX	+ 0.00830 * FUSEX	+ 0.2000 * _GIC_
6		:	-0.0566	<=	+ 0.1800 * FCNTX	+ 0.0157 * FGRIX	+ 0.1304 * FLPSX	+ 0.0277 * FMAGX	+ 0.7889 * FOSFX	+ 0.0193 * FUSEX		
7		:	0.1034	>=	+ 0.1800 * FCNTX	+ 0.0157 * FGRIX	+ 0.1304 * FLPSX	+ 0.0277 * FMAGX	+ 0.7889 * FOSFX	+ 0.0193 * FUSEX		
8		:	0.1000	>=								
9		:	-0.1000	<=	+ 0.00540 * FCNTX	+ 0.0339 * FGRIX	+ 0.00480 * FLPSX	+ 0.0294 * FOSFX	+ 0.00890 * FUSEX			
10		:	0.1000	>=	+ 0.00540 * FCNTX	+ 0.0339 * FGRIX	+ 0.00480 * FLPSX	+ 0.0294 * FOSFX	+ 0.00890 * FUSEX			
11		:	0.8470	<=	+ 1.0000 * CPO	+ 0.9333 * FCNTX	+ 0.8601 * FGRIX	+ 0.8037 * FLPSX	+ 0.9439 * FMAGX	+ 0.8746 * FOSFX	+ 0.9828 * FUSEX	
12		:	0.8670	>=	+ 1.0000 * CPO	+ 0.9333 * FCNTX	+ 0.8601 * FGRIX	+ 0.8037 * FLPSX	+ 0.9439 * FMAGX	+ 0.8746 * FOSFX	+ 0.9828 * FUSEX	

PROC NLP: Nonlinear Minimization

Null Space Method of Quadratic Problem

Parameter Estimates	9
Lower Bounds	9
Upper Bounds	9
Linear Constraints	12

Optimization Start

Active Constraints (+)	8	Objective Function	2.2247362956
Max Abs Gradient Element	4.9386144316		

Iter	Restarts	Function Calls	Active Constraints	Objective Function	Objective Function Change	Max Abs Gradient Element	Step Size	Slope of Search Direction
1	0	2	8'	1.53617	0.6886	2.5366	0.284	-2.828
2	0	3	8'	1.30289	0.2333	2.1082	0.127	-1.961
3	0	4	7'	0.49740	0.8055	1.2956	1.000	-1.611
4	0	5	6'	0.38178	0.1156	0.5746	1.000	-0.231
5	0	6	7	0.30634	0.0754	0.2703	0.490	-0.204
6	0	7	6'	0.28593	0.0204	0.3634	1.000	-0.0408
7	0	8	7	0.25601	0.0299	0.0424	0.396	-0.0943
8	0	9	6'	0.25586	0.000147	0.2336	1.000	-0.0003
9	0	10	6	0.25388	0.00198	2.22E-16	1.000	-0.0040

Optimization Results

Iterations	9	Function Calls	11
Gradient Calls	10	Active Constraints	6
Objective Function	0.2538751079	Max Abs Gradient Element	2.775558E-16
Slope of Search Direction	-0.003969247		

ABSGCONV convergence criterion satisfied.

PROC NLP: Nonlinear Minimization

Optimization Results
Parameter Estimates

N	Parameter	Estimate	Approx Std Err	t Value	Approx Pr > \|t\|	Gradient Objective Function	Active Bound Constraint
1	CPO	0.100000	0	9999.990000	0.000063662	9.079157	Equal BC
2	FBNDX	0	0	0	1.000000	0.624892	Lower BC
3	FCNTX	0	0	0	1.000000	-0.187342	Equal BC
4	FGRIX	0	0	0	1.000000	0.598914	Lower BC
5	FLPSX	0.107469	0.359891	0.298616	0.815262	0.100740	
6	FMAGX	0.354761	0.567678	0.624933	0.644415	0.282191	
7	FOSFX	0.094673	0.055857	1.694913	0.339340	-1.062665	
8	FUSEX	0.253954	0.583954	0.434887	0.738849	0.297033	
9	_GIC_	0.089143	0.248400	0.358869	0.780650	0.331132	

Value of Objective Function = 0.2538751079

Linear Constraints Evaluated at Solution

1 ACT -9.714E-17 = -1.0000 + 1.0000 * CPO + 1.0000 * FBNDX + 1.0000 * FCNTX + 1.0000 * FGRIX + 1.0000 * FLPSX + 1.0000 * FMAGX + 1.0000 * FOSFX + 1.0000 * FUSEX + 1.0000 * _GIC_

2 0.05623 = -0.0165 + 0.8603 * FBNDX + 0.00320 * FCNTX + 0.000300 * FGRIX + 0.0146 * FOSFX + 0.8000 * _GIC_

3 0.14377 = 0.2165 - 0.8603 * FBNDX - 0.00320 * FCNTX - 0.000300 * FGRIX - 0.0146 * FOSFX - 0.8000 * _GIC_

4 0.14163 = 0.0735 + 0.1397 * FBNDX + 0.0581 * FCNTX + 0.1057 * FGRIX + 0.1915 * FLPSX + 0.0561 * FMAGX + 0.0814 * FOSFX + 0.00830 * FUSEX + 0.2000 * _GIC_

5 0.05837 = 0.1265 - 0.1397 * FBNDX - 0.0581 * FCNTX - 0.1057 * FGRIX - 0.1915 * FLPSX - 0.0561 * FMAGX - 0.0814 * FOSFX - 0.00830 * FUSEX - 0.2000 * _GIC_

6 0.16000 = 0.0566 + 0.1800 * FCNTX + 0.0157 * FGRIX + 0.1304 * FLPSX + 0.0277 * FMAGX + 0.7889 * FOSFX + 0.0193 * FUSEX

7 ACT 4.5103E-17 = 0.1034 - 0.1800 * FCNTX - 0.0157 * FGRIX - 0.1304 * FLPSX - 0.0277 * FMAGX - 0.7889 * FOSFX - 0.0193 * FUSEX

8 0.10000 = 0.1000

9 0.10556 = 0.1000 + 0.00540 * FCNTX + 0.0339 * FGRIX + 0.00480 * FLPSX + 0.0294 * FOSFX + 0.00890 * FUSEX

10 0.09444 = 0.1000 - 0.00540 * FCNTX - 0.0339 * FGRIX - 0.00480 * FLPSX - 0.0294 * FOSFX - 0.00890 * FUSEX

11 0.00658 = -0.8470 + 1.0000 * CPO + 0.9333 * FCNTX + 0.8601 * FGRIX + 0.8037 * FLPSX + 0.9439 * FMAGX + 0.8746 * FOSFX + 0.9828 * FUSEX

12 0.01342 = 0.8670 - 1.0000 * CPO - 0.9333 * FCNTX - 0.8601 * FGRIX - 0.8037 * FLPSX - 0.9439 * FMAGX - 0.8746 * FOSFX - 0.9828 * FUSEX

RHS
0.253875

Iteration 3 minobjfn = 0.2499677559 last objfnval = 0.2499677559

RHS
0.249968

This Page is Inserted by IFW Indexing and Scanning Operations and is not part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☑ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☑ FADED TEXT OR DRAWING
- ☐ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☐ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: _____

IMAGES ARE BEST AVAILABLE COPY.
As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.